

Grzegorz Murzynowski

Technical Remarks Concerning  
the Anotatornia Service

The **NKJP** Version v.1.180

2nd July 2010



# Contents

## INTRODUCTION 7

1. Installation and deployment 7
  - 1.1. The Transza's—creation and allocation 8
2. Multi-level annotation 9
  - 2.1. Statuses 10
  - 2.2. Reports 12
  - 2.3. Editability 12
  - 2.4. Handling of word-level segmentation ambiguities 12
  - 2.5. Morphosyntactic annotation 12
3. Export of annotated data 13
4. Path and Transza's finder 13
5. See also 13
6. Contact 13

## THE TABLES' DEFINITIONS 15

## THE RUBY AND RHTML CODE 27

- app/controllers 27
  - admin\_controller.rb 27
  - anotacja\_controller.rb 29
  - application.rb 67
  - aux\_controller.rb 72
  - debug\_controller.rb 73
  - logowanie\_controller.rb 74
  - superanotacja\_controller.rb 76
  - zarzadca\_controller.rb 87
- app/helpers 88
  - admin\_helper.rb 88
  - anotacja\_helper.rb 88
  - application\_helper.rb 89
  - debug\_helper.rb 91
  - logowanie\_helper.rb 91
  - migration\_helper.rb 91
  - zarzadca\_helper.rb 93
  - zdanie\_helper.rb 94
- app/models 94
  - akapit.rb 94

- akapit\_transzy.rb 108
- cz\_m.rb 132
- cz\_m\_leksem.rb 134
- dummy1.rb 136
- dummy2.rb 136
- fraza\_anot.rb 136
- fraza\_typ.rb 141
- interpretacja.rb 144
- interpretacja\_anot.rb 150
- interpretacja\_export.rb 150
- interpretacja\_z\_leksemem.rb 151
- klasa\_gram.rb 152
- komentarz.rb 154
- koniec\_zdania\_anot.rb 155
- leksem.rb 156
- morph.rb 157
- morphosyntactic\_rozbieznosc.rb 158
- nowa\_segmentacja.rb 158
- path.rb 164
- poziomy\_anotacji.rb 165
- prosby\_anotatorek.rb 173
- protokol.rb 175
- punkt\_protokolu.rb 178
- ramkowanie.rb 179
- rola.rb 187
- segmentation\_rozbieznosc.rb 190
- sens\_anot.rb 190
- sensy.rb 191
- sensy\_leksemu.rb 194
- sensy\_sensem.rb 194
- sentences\_rozbieznosc.rb 195
- sg\_choice.rb 195
- sg\_variant.rb 196
- sg\_variant\_anot.rb 198
- statusy.rb 199
- tagset.rb 204
- token.rb 213
- transza.rb 222
- uzytkownik.rb 226
- word\_senses\_rozbieznosc.rb 231
- app/views/admin 232
  - dodaj\_uzytkownika.rhtml 232
  - lista\_uzytkownikow.rhtml 233
  - usun\_uzytkownika.rhtml 234

zablokuj\_uzytkownika.rhtml 234  
 zmien\_dane.rhtml 234  
 zmien\_haslo.rhtml 235  
 app/views/anotacja 236  
   \_akapity\_skomentowane.rhtml 236  
   \_annotuj\_zatwod.rhtml 237  
   \_annotuj\_zatwod\_nk.rhtml 237  
   \_blizniaczy\_status.rhtml 239  
   \_fra\_sie\_typ.rhtml 239  
   \_fra\_wyb\_glowy.rhtml 240  
   \_komentarz\_ballot.rhtml 240  
   \_komentarz\_dod\_link.rhtml 241  
   \_komentarz\_dodaj.rhtml 241  
   \_komentarze.rhtml 242  
   \_lista\_akapitow.rhtml 243  
   \_lista\_odebranych.rhtml 244  
   \_lista\_transz.rhtml 245  
   \_nowa\_segmentacja.rhtml 246  
   \_odbierz\_transze.rhtml 247  
   \_ogladactwo\_goto.rhtml 248  
   \_ogladactwo\_lista\_akapitow.rhtml 248  
   \_prosba\_link.rhtml 249  
   \_prosby\_annotatorek.rhtml 249  
   \_przyklady\_sensow\_link.rhtml 252  
   \_quasi\_transza.rhtml 252  
   \_sens\_long.rhtml 252  
   \_sens\_pelny\_opis\_link.rhtml 253  
   \_sentenśnik.rhtml 254  
   \_status\_info.rhtml 254  
   \_token\_disamb.rhtml 254  
   \_token\_fra.rhtml 256  
   \_token\_framka.rhtml 262  
   \_token\_segm.rhtml 262  
   \_token\_sens.rhtml 263  
   \_token\_td.rhtml 265  
   \_token\_tworz\_tag.rhtml 268  
   \_token\_wszi.rhtml 270  
   \_token\_wybs.rhtml 271  
   \_tworz\_tag\_lemat.rhtml 272  
   \_wsze\_transze\_BW.rhtml 272  
   \_wsze\_transze\_otwarte.rhtml 274  
   \_wsze\_transze\_otwarte\_BW.rhtml 275  
 anotuj.rhtml 276  
 autocomplete\_tag.rhtml 280  
 lista\_transz.rhtml 280  
 nie\_chce\_transzy.rhtml 281  
 ogladactwo.rhtml 282  
 pokaz\_transze.rhtml 283  
 quasi\_transza\_brutal.rhtml 283  
 sense\_inventory.rhtml 284  
 wyszukaj\_akapity.rhtml 284  
 zweryf.rhtml 285  
 app/views/aux 286  
   sensy.rhtml 286  
 app/views/debug 287  
   \_lista\_sciezek.rhtml 287  
   \_lista\_transz.rhtml 287  
   \_mwa\_ha\_ha.rhtml 288  
 debug-trash.rhtml 288  
 debug.rhtml 293  
 elektroforeza.rhtml 293  
 app/views/layouts 295  
   \_bug\_reports.rhtml 295  
   \_notice.rhtml 295  
   \_powitalny.rhtml 295  
   \_sidebary.rhtml 296  
 admin.rhtml 297  
 zdanie.rhtml 298  
 app/views/logowanie 299  
   \_session\_expiry.rhtml 299  
 dodaj\_uzytkownika.rhtml 299  
 index.rhtml 300  
 lista\_uzytkownikow.rhtml 300  
 logowanie.rhtml 300  
 usun\_uzytkownika.rhtml 301  
 wylogowanie.rhtml 301  
 zablokuj\_uzytkownika.rhtml 301  
 app/views/zarzadca 302  
   lista\_annotatorow.rhtml 302  
   odbierz\_transze.rhtml 302  
   przypisz\_transze.rhtml 303  
 db/migrate 303  
   001\_add\_updated.rb 303  
   002\_komentarz\_add\_nowy.rb 304  
   003\_komentarz\_index\_created.rb 305  
   004\_token\_add\_kolejnosc.rb 305  
   006\_sg\_choice\_variant\_add\_dodany.rb 306  
   007\_statusy\_add\_akapit\_id.rb 306  
   008\_token\_add\_superancje.rb 307  
   009\_spatialize\_npses.rb 307  
   010\_reverse\_niedostrzeganiem\_masturbacji.rb 307  
   011\_add\_sensowe.rb 308  
   012\_nowa\_segmentacja\_add\_nps.rb 309  
   013\_dummy\_table.rb 310  
   014\_przeklnij\_morphosynty.rb 310  
   015\_debug\_potworne\_czasy.rb 313  
   016\_przyspiesz\_danowlew.rb 313  
   017\_popraw\_lex\_xmlidy.rb 313  
   018\_przepisz\_baze.rb 314  
   019\_popraw\_tok\_xpointers.rb 323  
   020\_popraw\_komentarze.rb 325  
   021\_add\_uzytkownik\_tylko\_nowe\_komentarze.rb 325  
   022\_add\_dziubnij\_w\_segmentacjach.rb 325  
   023\_popraw\_xpointer\_przecinek.rb 326  
   024\_dziubnij\_w\_segmentacjach.rb 327  
   025\_dziubnij\_w\_segmentacjach.rb 327  
   026\_zasegmentuj\_dolewke.rb 328  
   027\_popraw2968.rb 329

o28_cz_m_leksem_id_indices.rb	329	o43_popraw_segmi2946.rb	338
o30_sens_anot_dopisz_token_id.rb	330	o44_token_index_orth.rb	339
o31_add_czmleksem_dopisany.rb	331	o45_popraw_intrj.rb	339
o32_create_wsd_rozbieznosc.rb	331	o46_popraw_sierzanty.rb	339
o33_sensy_add_opishtml.rb	331	o47_popraw_interpretacja_leksem.rb	340
o34_statusy_index_akapitid.rb	332	o48_nie_przypisane_sensy.rb	342
o35_akapit_add_tresc.rb	332	o49_popraw_polpauze.rb	342
o36_statusy_add_transzaid.rb	332	o50_popraw_xpointery.rb	343
o37_statusy_popraw_wsd.rb	333	o51_add_interpretacja_export.rb	344
o38_uspojnij_path_id.rb	333	o52_popraw_segmi10391.rb	344
o39_popraw_dwa_dywizy.rb	336	o53_xpointers_not_null.rb	345
o40_uzupelnij_dolewke.rb	336	o54_popraw_dziabag_ta.rb	345
o41_revert_autowsd.rb	337	o55_popraw_sg_nom.rb	346
o42_popraw_nierowne_statusy.rb	337		



# Introduction

This is Anotatornia, a Ruby on Rails-based application for manual multi-level annotation of texts.

The Anotatornia program, i.e. the contents of directories `app/`, `db/` and files contained in the main directory, are released under the **GNU** Public License version 3.

A compatible copy of the Ruby on Rails framework, i.e. the contents of the `vendor/rails/` directory, is provided under the **MIT** license.

For the texts of those licenses see file **COPYING**.

However, if you don't want to read these licenses, please note at least that both of them claim that

**This software is provided free of charge and free to be used in almost any manner but AS IS, i.e. WITHOUT ANY GUARANTEE OF ANY KIND (such as: that it works, won't destroy your data and so on).**

## 1. Installation and deployment

Anotatornia has been installed only on Linux machines (Ubuntu and Red Hat Enterprise).

To make Anotatornia work you'll need:

- Ruby v.>=1.8,
- SQLite3 v.>=3.3.6
- sqlite3-ruby v.>=1.2.4
- mongrel v.>=1.1.5
- Rails v==1.2.4 (provided in this package)

As you are reading this file, let's assume you have unpacked the contents of the archive to a directory `~/anotatornia` (and you want it to stay there). To start, you should

```
sqlite3 <database-name>.db
.read schema_20106005.sql
<C-d (quit SQLite3)>
```

This creates the database file and gives it Anotatornia's structure. Since in the **NKJP** project there were a couple of linguistic databases with the same set of users, the users' database is contained in another file. Create it:

```
sqlite3 <users_db>.db
.read schema_u.sql
<C-d (quit SQLite3)>
```

Now configure the Rails code: add a stanza for your database to the `config/database.yml` file and define an environment file for it (named the same as the stanza in `config/database.yml`).

To be sure the database is most up-to-date run also

```
RAILS_ENV=<your env. name> rake db:migrate
```

Then you can probably input linguistic data. The morphosyntactic tagset is in the file `gramdata/tagset.cfg` and will be automatically loaded to the table `tagset` at the first startup of the application.

Because of historical reasons (word-sense disambiguation was deployed after deployment of the first three levels) input of the sense dictionary is not automatic.

The senses dictionary is contained in the file `nkjp_wsd_si/NKJP_WSI.xml`.

To input it you should change the datetime in line 46 in the Ruby script `isens.rb` (contained in the main directory) to some future value (it's a sentinel not to destroy the senses once input). Then you launch the script in the proper environment, which is done by typing in a terminal:

```
script/console nk8005
load 'isens.rb'
```

You can replace the senses dictionary until word-sense disambiguation started only.

You can add new users via the admin interface. During the very first launch of a new Anotatornia a default admin is created with login 'QueenOfSpades' and password '15 mars 1615 Paris'. It is strongly advisable that you delete this user after creating your own admin(s) or at least change her password.

The texts should be in **NKJP XML** format. Then they can be input with the `danowlew` shell script:

```
danowlew <envir> <path to texts>
```

That script launches `im.rb` which in turn inputs the data into the proper database and initializes it, i.e. creates *akapit transzy's*—'a bulk's paragraphs', i.e. groups of Akapit's (paragraphs) destined to be attributed at once (together), explained in the next section.

To make your Anotatornia available from outside you should open some port (in **NKJP** project it was 8003) and write it down in the `anotatornia-mongrels` script. Then run

```
./anotatornia-mongrels start
```

for single launch (during tests etc.) or

```
./anotatornia-keeprunning
```

for full deployment. The latter script launches mongrel server(s) as set in the file `anotatornia-mongrels` and every 10 seconds checks whether are they running.

### 1.1. The Transza's—creation and allocation

A unit of text to be worked on is Akapit, a paragraph that is. Each Akapit should be annotated by two Annotators. Akapits have to be allocated in such a manner that co-learning of the Annotators is minimised.



Akapit's are allocated to the Annotators in Transza's ('bulks'). The number of Akapit's in a Transza is determined by `AnoVersion.rozmiar_transzy` (defined in `config/environments/<env>.file`) and for the deployment in **NKJP** was 10.

To make a Transza consisting of 10 Akapits every next 20 of Akapits is divided in fives and those fives are paired in different ways to allocate them to four different Annotators. Let's consider Akapit's 1–20:

- Transza 1: Akapit's 1–5, 6–10 (fives 1 & 2);
- Transza 2: Akapit's 1–5, 11–15 (fives 1 & 3);
- Transza 3: Akapit's 11–15, 16–20 (fives 3 & 4);
- Transza 4: Akapit's 6–10, 16–20 (fives 2 & 4).

From such a four of intersecting Transza's any Annotator can get at most one (therefore the minimum number of Annotators required in Anotatornia is 4).

The Transza's are allocated at Annotator's request in the order of creation, which is according to the order of inputting of the data into the database, which in particular is according with the order of Akapit's.

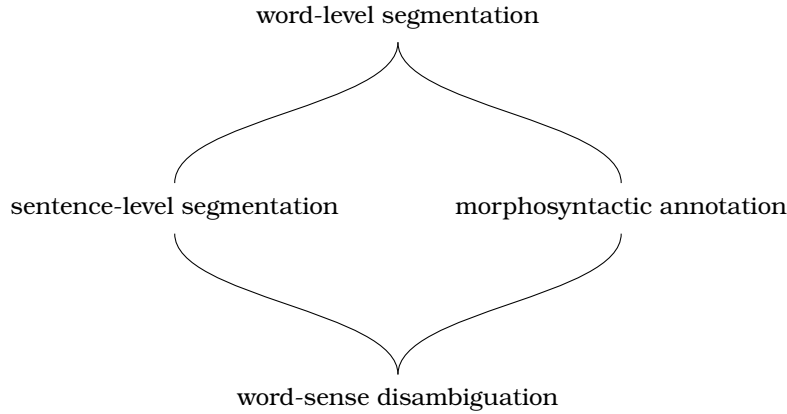
Note that it is perfectly consistent with the above conditions that Annotator *X* gets the Transza's 1, 8, 9, 14, 17.

## 2. Multi-level annotation

Anotatornia provides four levels of annotation:

- word-level segmentation,
- sentence-level segmentation,
- morphosyntactic annotation,
- word-sense disambiguation.

Since the sentence limits cannot be determined before determining the limits of words and so on, the levels constitute a partial order that can be described with a following diagram:



which is also encoded in the model file `poziomy_annotacji.rb`. Annotation at level *x* is possible (status o, see next section), if

$$(\forall_{y \in \text{Levels}} (\text{succ}(y) = x \rightarrow \text{Verified}(y))) \wedge \neg \text{Closed}(x)$$

It is clear that at the minimal level (word-level segmentation) the first part of the conjunction is empty-satisfied.

If there is no word-level segmentation ambiguity in an Akapit, it is automatically “empty-verified” at this level.

When an Annotator enters the annotation of an Akapit, they is shown available levels and their immediate precedents. For a “fresh” Akapit it usually will be the sentence-level segmentation and morphosyntactic annotation level (word-level segmentation ambiguities are rather rare). The Annotator can choose which levels should be displayed. According to this the buttons “Zatwierdź *<level>*” (‘Commit ...’) are displayed and, if there are more than one levels, “Zatwierdź wszystkie” (‘Commit all’).

Clicking a “Zatwierdź *<level>*” button launches a subroutine checking correctness of annotation at this level and, if the test turns true, change of the Akapit’s status to 8, :zatwierdzone (‘ratified’), or, if the counter-Annotator has annotated it too, to 16, :zweryfikowane (‘verified’) or 10 :do\_poprawki (‘to be fixed’).

In the latter case the Akapit is returned to the Annotator to be looked at again. The points of Annotator’s disagreement are highlighted but annotation of the counter-Annotator is not shown.

After another commitment of the Akapit it is checked again and, if the annotations fully agree, it’s marked as verified (status 16), otherwise it’s marked :do\_osądzania (‘to be judged’) (status 14).

(In case the counter-Annotator didn’t re-annotate the Akapit, it gets status 12, :po\_poprawce (‘after the fix’) and the above routine will be applied to the “fixing” commitment of the Akapit by the counter-Annotator.)

The Akapit’s with status  $\geq 14$  (:do\_osądzania) can be annotated only by a Super-Annotator. With making any slightest change by a Super-Annotator (which may occur at any level not higher than than maximal allowed) the Akapit’s status at this level changes into 15, :osądzane (‘under judgment’), and at all the levels higher into -1, :nie\_dopuszczone (‘not allowed’).

An annotating user can change the Akapit’s annotation at any available level, at which the Akapit is not yet verified. Making a change in a ‘committed’ Akapit results with degrading its status (at this level) to 0, :dopuszczony (‘allowed’), if the user is an Annotator, or 14, :osądzany, if the user is a Super-Annotator.

## 2.1. Statutes

Since we divide annotation into levels, we assign a level-status to each Akapit Transzy (‘paragraph of a bulk’), i.e. a pair

*<akapit, anotator>*:

- 1 (the default) Akapit Transzy is nor annotated neither allowed to be annotated at this level;
- 0 A.T. is allowed to be annotated at this level;
- 8 A.T. has been annotated and is ready for confrontation with its twin;
- 10 to be fixed: verifying subroutine was launched at this Akapit for the first time and found disagreement(s), the Akapit is returned to Annotators.
- 16 A.T. has been verified;
- 14 to be judged: the verifying subroutine found disagreement(s), the Akapit is passed to Super-Annotators;
- 15 under judgement: a Super-Annotator made some change to the annotation but has it not yet committed;
- 18 judged: the Akapit has been annotated by a Super-Annotator.

In the model file `akapit_transzy.rb` there are defined (in the clause `if DlaEli.nk`) status methods referring to the record of the table `statusy` connected with given Akapit Transzy.

Because of four levels of annotation the answer to the question “How many Akapit’s have been annotated” becomes a vector and so it appears at the main site after logging in, next to bulks in a list and so on.

When a status at a given level should be increased, the method `anAnotacjaController.podnies_status` (‘increase status’) is called. It looks at the hash `STATI_podnies` hardcoded in the model `akapit_transzy.rb`, in which the graph of status increasing is encoded. The `stati` that will be increased independent of the verification result, appear in that hash `sauté`. Those whose increasing depends of the verification result are encoded as an array [`<status>`, `<werdykt>`].

An assumption is made that there will not be a verification (the object `session[:weryfikacja]`) for those `stati` that don’t need to be verified.

A method is defined `aStatusy.setpoziom` as a shorthand for a thing I did in two places, i.e. passing to an object of the class `Statusy` the method `<poziom>=`.

Each level of annotation has in the table `statusy` the column `<level name>_uzid` for the user who committed that level. We assume that they usually are an Anotator, in whose bulk (Transza) this Akapit is present. But they also can be a Super-Anotator, and in case of word-level segmentation also an Admin (who changes something “by hand”).

In the views `anotuj.rhtml`, `_token_disamb.rhtml`, `_token_sens.rhtml`, `_token_fra.rhtml` (and `_walencja.rhtml`, not used in the **NKJP** version) a helper’s method is used `zt_edytowalne?`, defined in the helper `anotacja_helper.rb`. This method in the previous version of Anotatoria checked whether an Akapit is available for editing by the given user at the given status. In the **NKJP** project, where we distinguish annotation levels, this method has to take into account also them, i.e. to check availability with three respects: user, level and status.

Another thing is the question whether the given annotation level should be displayed. It seems however reasonable to make the method `zt_edytowalne?` return **false** for the levels hidden at the moment, so that making changes to the annotation at the hidden levels would be forbidden.

This method for **NKJP** has to be completely rewritten, two-argument not one-. Therefore it seems reasonable to create separate versions of those views for the Project, to write `zt_edytowalne?` with proper arity in each version.

So, the method `zt_edytowalne?` in the **NKJP** version is two-variant: if called with one argument, which in such case has to be a user **ID**, then it checks availability of given Akapit for *any* editing (annotation): whether the user is permitted to annotate and whether the Akapit has an unfinished level and allowed for annotation. If called with two arguments, which have to be `<<level>`, `<user ID>`, the method checks editability at this given level.

Besides the level-`stati` (stored in the table `statusy`) we also use the “grand” status: the column `status` of the table `akapit_transzy` which changes iff the minimal status of a maximal level gets changed. To handle this a method is added `self.maksymalne` in the model `poziomy_anotacji.rb`.

## 2.2. Reports

Occurrence of a disagreement between the two annotations of a given Akapit is recorded in the tables `protokol` ('report'), `punkt_protokolu` ('report's item') and `<level>_rozbieznosc` ('disagreement'). Also agreeing upon a word-level segmentation disambiguation is reported, in the table `segmentation_protokol`. The table `punkt_protokolu` is polymorphically associated with `segmentation_protokol` and with the tables `<poziom>_rozbieznosc`. Handling of such an association is provided by Rails themselves.

## 2.3. Editability

The filter `anApplication.check_edytowalne` ('check the editables') looks at the grand status of the given Akapit Transzy (the field status of the table `akapit_transzy`) and whether the parameters require `:poziom`, and checks editability at that level if so.

```
AkapitTranszy.editable_par_annotate?  
anAkapitTranszy.edytowalne_przez_anotatora?( poziom )  
anAnotacjaHelper.zt_edytowalne?( * args ), where * args are [ <user ID> ] or  
[ <user ID>, :<level>]
```

We check editability twice: first time while rendering a view, to hide the links to the unavailable actions, and second time before executing an action, in the filter `check_edytowalne`, so that the action be really unavailable.

## 2.4. Handling of word-level segmentation ambiguities

The source corpus data for Anotatornia were output of Morfeusz. Therefore the word-level segmentation ambiguities were given as a disjunction: two tokens—one token (concatenated), and each part of this disjunction has its own set of morphosyntactic interpretations (possible tags).

With a huge amount of work also a possibility of adding new word-level segmentations was implemented, still not perfect since the pathological possibilities are endless.

## 2.5. Morphosyntactic annotation

For each token (word segment) there is a list of possible morphosyntactic tags computed by Morfeusz and input as a part of the source data.

Due to the **NKJP** project needs there is also implemented a possibility of creating (by the Annotators) new morphosyntactic tags for the tokens. Such a new tag is checked against the tagset input at each startup of Anotatornia from the file `gramdata/tagset.cfg` (in the Poliqarp format).

The tagset table is polymorphic in a sense: the `tagset.cfg` file consists of a couple of parts (attributes, grammatical classes etc.) and the table stores them as serialised Ruby objects, which, thanks to built-in Rails methods, makes access to them seamless: the lists of attributes are serialised as arrays, attribute values as Regexp (disjunctional) etc.

In the class `Tagset`, being a Rails frame for the tagset table, a method is defined `Tagset.check_tag( tag_string )`, which returns `[true]` if `tag_string` is a correct tag or else `[false, <what's wrong with it>]`.

At the annotation page (anotuj.rhtml, \_token\_wszl.rhtml, \_token\_disamb.rhtml) we implement a list field for choosing the tag from those computed by Morfeusz and a text field for adding a new tag by Anotator. The latter is equipped with auto-completion which should help to create a correct tag (anyway it'll be checked when committed).

### 3. Export of annotated data

You export the annotated data with the `sowa` shell script (it uses `exml.rb`):

```
./sowa [<argument>]
```

The *<argument>*s accepted by `sowa` are intended to narrow the set of data to export and are described in the script. They include Ruby's number interval (for interval of Akapit's) and a Ruby's (almost Perl's) Regexp to specify the Corpus path(s).

If no argument is provided then all the annotated Akapit's are exported.

You should modify that script to make it copy and export *your* database.

### 4. Path and Transza's finder

Because of the need of easy finding the Corpus path or the Transza of a given Akapit a view is provided `debug/elektroforeza`.

### 5. See also

The zip archive you got (as we assume since you read this file) contains a Dia file `anotatornia.dia` containing a diagram of the database.

For the (linguistic) details of annotation see also `instrukcja_NK.pdf` (in Polish).

There's also a presentation (in `le.pdf`) *Manual annotation of the National Corpus of Polish with Anotatornia* by Adam Przepiórkowski and me (Grzegorz Murzynowski) at `nlp.ipipan.waw.pl/Anotatornia/anotatornia.pdf`

and an article of the same title and authors in Proceedings of the **PALC 2009** Conference, published by Peter Lang, Frankfurt am Main, ed. by Stanisław Goźdz-Roszkowski.

### 6. Contact

About technical (implementation) matters of Anotatornia please contact Grzegorz Murzynowski, `natror<at>o2<dot>p1`.

About other matters concerning Anotatornia and the **NKJP** project please contact Adam Przepiórkowski, Ph.D., Associate Professor, `Adam.Przepiorkowski<at>ipipan<dot>waw<dot>p1`.



## The tables' definitions

```
create table akapit(  
    akapit_id Integer primary key autoincrement,  
    morph_id Integer not Null,  
    path_id Integer not Null, -- używane przy wyszukiwaniu akapitu  
    podczas wczytywania sg.xml.  
    segmentation_xmlid Text,  
    segmentation_xlink_href Text,  
    morphosyntactic_xmlid Text,  
    created_at TimeStamp,  
    updated_at TimeStamp  
    , tresc Text, incipit Text);  
  
create table akapit_transzy(  
    akapit_transzy_id Integer primary key autoincrement,  
    akapit_id Integer not Null,  
    transza_id Integer not Null,  
    status Integer default 0 not Null,  
    odrzucony Boolean default 'f',  
    uzytkownik_id Integer,  
    created_at TimeStamp,  
    updated_at TimeStamp,  
    blizniaczy_id Integer);  
  
create table cz_m(  
    -- Attributes --  
    cz_m_id Integer primary key autoincrement,  
    cz_m_ozn Text,  
    created_at TimeStamp,  
    updated_at TimeStamp);  
  
create table cz_m_leksem(  
    -- Attributes --  
    cz_m_leksem_id Integer primary key autoincrement,  
    lemat Text not Null,  
    xmlid Text not Null,  
    cz_m_id Integer not Null,  
    created_at TimeStamp,  
    updated_at TimeStamp);  
  
create table fraza_typ(  
    -- Attributes --  
    fraza_typ_id Integer primary key autoincrement,  
    typ_opis Text,
```

```

typ_symbol Text,
synhead_list Text,
semhead_list Text,
jednoglowa Boolean,
created_at TimeStamp default current_timestamp,
updated_at TimeStamp);

create table interpretacja(
    interpretacja_id Integer primary key autoincrement,
    token_id Integer not Null,
    lex_xmlid Text not Null,
    msd_xmlid Text not Null,
    path_id Integer not Null,
    numer_lex_token Integer,
    reszta_tagu Text,
    leksem_id Integer,
    disamb Boolean default 'f',
    dodana Boolean default 'f',
    cz_m_leksem_id Integer,
    sensy_id Integer,
    created_at TimeStamp,
    updated_at TimeStamp , cz_m_leksem_przypisany Boolean default
    'f');

create table interpretacja_anot (
    interpretacja_anot_id Integer primary key autoincrement,
    akapit_transzy_id Integer not Null,
    uzytkownik_id Integer not Null,
    token_id Integer not Null,
    interpretacja_id Integer not Null,
    created_at TimeStamp,
    updated_at TimeStamp);

create table interpretacja_bef47_bad_leksem( interpretacja_id
Integer primary key, leksem_id Integer );

create table interpretacja_bef47_bad_xmlids( interpretacja_id
Integer primary key, lex_xmlid Text, msd_xmlid Text,
numer_lex_token Integer );

create table klasa_gram(
    klasa_gram_id Integer primary key autoincrement,
    cz_m_id Integer,
    klasa_gram_nazwa Text,
    klasa_gram_ozn Text,
    klasa_sem Text,
    created_at TimeStamp,
    updated_at TimeStamp);

create table komentarz(
    komentarz_id Integer primary key autoincrement,
    akapit_id Integer,
    uzytkownik_id Integer,

```



```
akapit_transzy_id Integer,  
przyczyna_odrzucenia Integer default 0,  
tresc Text,  
nowy Boolean default 't',  
created_at TimeStamp,  
updated_at TimeStamp);  
  
create table koniec_zdania_anot(  
    koniec_zdania_anot_id Integer primary key autoincrement,  
    akapit_id Integer not Null,  
    akapit_transzy_id Integer not Null,  
    uzytkownik_id Integer not Null,  
    token_id Integer not Null,  
    created_at TimeStamp,  
    updated_at TimeStamp  
);  
  
create table leksem(  
    leksem_id Integer primary key autoincrement,  
    lex_xmlid Text not Null,  
    path_id Integer not Null, -- to ma sens: do utożsamienia z eltem <fs  
    type="lex"> w Korpusie.  
    lemat Text,  
    klasa_gram_id Integer,  
    cz_m_id Integer, -- do sensowania (redundantne, o+)  
    cz_m_leksem_id Integer, -- do sensowania (o+)  
    created_at TimeStamp,  
    updated_at TimeStamp, cz_m_przypisana Boolean default 'f',  
    cz_m_leksem_przypisany Boolean default 'f');  
  
create table morph(  
    morph_id Integer primary key autoincrement,  
    xmlid Text not Null,  
    path_text Text not Null, -- identical as in table path  
    created_at TimeStamp,  
    updated_at TimeStamp  
);  
  
create table morphosyntactic_rozbieznosc(  
    morphosyntactic_rozbieznosc_id Integer primary key  
    autoincrement,  
    token_id Integer not Null,  
    interpretacja_id Integer not Null,  
    reszta_tagu Text,  
    leksem_id Integer not Null,  
    created_at TimeStamp,  
    updated_at TimeStamp  
);  
  
create table nowa_segmentacja(  
    nowa_segmentacja_id Integer primary key autoincrement,  
    ids Text,
```

```

total_akapitow Integer, -- == (select count(*) from akapit) ensured
by a trigger
seg_is Text,
seg_should_be Text,
nps Boolean not Null default 't',
wprowadzona Boolean default 'f'
, akapit_transzy_id Integer, transza_id Integer);

create table path(
    path_id Integer primary key autoincrement,
    path_text Text not Null,
    created_at TimeStamp,
    updated_at TimeStamp
);

create table poziomy_annotacji(
    poziomy_annotacji_id Integer primary key autoincrement,
    uzytkownik_id Integer not Null,
    segmentation Boolean default 'f',
    sentences Boolean default 'f',
    morphosyntactic Boolean default 'f',
    word_senses Boolean default 'f',
    syntactic_words Boolean default 'f',
    named_entities Boolean default 'f',
    syntactic Boolean default 'f',
    created_at TimeStamp,
    updated_at TimeStamp);

create table prosby_annotatorek(
    prosby_annotatorek_id Integer primary key autoincrement,
    dotyczy_id Integer not Null,
    dotyczy_type Text not Null,
    uzytkownik_id Integer not Null,
    opis Text,
    created_at TimeStamp,
    updated_at TimeStamp,
    rozpatrzona Boolean default 'f',
    spelniona Boolean);

create table protokol(
    protokol_id Integer primary key autoincrement,
    poziom Text not Null,
    akapit_id Integer not Null,
    czy_rozbieznosc Boolean not Null default 't',
    czy_superancja Boolean not Null default 'f',
    created_at TimeStamp,
    updated_at TimeStamp
);

create table punkt_protokolu(
    punkt_protokolu_id Integer primary key autoincrement,
    elt_protokol_id Integer not Null,

```

```

        elt_protokol_type Text not Null,
        protokol_id Integer not Null,
        akapit_transzy_id Integer not Null,
        uzytkownik_id Integer,
        created_at TimeStamp,
        updated_at TimeStamp);

create table rola(
        opis Text,
        rola_id Integer primary key,
        opis_kr string,
        created_at TimeStamp,
        updated_at TimeStamp);

create table schema_info (version Integer);

create table segmentation_rozbieznosc(
        segmentation_rozbieznosc_id Integer primary key autoincrement,
        sg_choice_id Integer not Null,
        sg_variant_id Integer not Null,
        created_at TimeStamp,
        updated_at TimeStamp
);

create table sens_anot(
-- Attributes --
        sens_anot_id Integer primary key autoincrement,
        uzytkownik_id Integer,
        akapit_transzy_id Integer not Null,
        token_id Integer not Null,
        interpretacja_id Integer not Null,
        sensy_id Integer not Null,
        automatycznie Boolean,
        created_at TimeStamp, -- default current_timestamp,
        updated_at TimeStamp);

create table sensy(
-- Attributes --
        sensy_id Integer primary key autoincrement,
        cz_m_id Integer not Null,
        cz_m_leksem_id Integer not Null,
        xmlid Text,
        n Integer, -- przy wlewie sprawdzamy czy lemat.odpolszcz.n == xmlid
        short_def Text,
        long_def_xml Text,
        long_def_html Text,
        created_at TimeStamp,
        updated_at TimeStamp
);

create table sentences_rozbieznosc(
        sentences_rozbieznosc_id Integer primary key autoincrement,
        token_id Integer not Null,

```

```

        czy_konczy_zdanie Boolean not Null,
        created_at Timestamp,
        updated_at Timestamp
    );

create table session(
    "id" Integer primary key not Null,
    "session_id" varchar(255) default Null,
    "data" Text default Null,
    "updated_at" datetime default Null);

create table sg_choice(
    sg_choice_id Integer primary key autoincrement,
    akapit_id Integer not Null, -- redundantne, dla łatwiejszego
    wyszukiwania
    dodany Boolean default 'f',
    xmlid Text,
    created_at Timestamp,
    updated_at Timestamp
    );

create table sg_variant(
    sg_variant_id Integer primary key autoincrement,
    sg_choice_id Integer not Null,
    xmlid Text,
    dodany Boolean default 'f',
    created_at Timestamp,
    updated_at Timestamp
    );

create table sg_variant_anot(
    sg_variant_anot_id Integer primary key autoincrement,
    sg_choice_id Integer not Null,
    sg_variant_id Integer not Null,
    chosen Boolean not Null,
    akapit_transzy_id Integer not Null,
    uzytkownik_id Integer,
    created_at Timestamp,
    updated_at Timestamp
    );

create table statusy(
    statusy_id Integer primary key autoincrement,
    akapit_transzy_id Integer not Null,
    akapit_id Integer not Null,
    segmentation Integer default -1,
    sentences Integer default -1,
    morphosyntactic Integer default -1,
    word_senses Integer default -1,
    syntactic_words Integer default -1,
    named_entities Integer default -1,
    syntactic Integer default -1,

```

```

segmentation_uzid Integer,
sentences_uzid Integer,
morphosyntactic_uzid Integer,
word_senses_uzid Integer,
syntactic_words_uzid Integer,
named_entities_uzid Integer,
syntactic_uzid Integer,
segmentation_odrzucony Boolean default 'f',
sentences_odrzucony Boolean default 'f',
morphosyntactic_odrzucony Boolean default 'f',
word_senses_odrzucony Boolean default 'f',
syntactic_words_odrzucony Boolean default 'f',
named_entities_odrzucony Boolean default 'f',
syntactic_odrzucony Boolean default 'f',
created_at Timestamp,
updated_at Timestamp, transza_id Integer);

create table tagset(
    tagset_id Integer primary key autoincrement,
    typ Text,
    lewe Text,
    prawe Text,
    created_at Timestamp,
    updated_at Timestamp);

create table token(
    token_id Integer primary key autoincrement,
    kolejnosc Integer, -- not null ensured by a trigger
    akapit_id Integer not Null,
    xpointer Text,
    segmentation_xmlid Text,
    morphosyntactic_xmlid Text,
    fs_morph_comment Text,
    path_id Integer not Null,
    orth Text,
    czy_interp Boolean not Null,
    ns_poprzedza Boolean default 'f',
    ns_nastepuje Boolean,
    czy_konczy_zdanie Boolean,
    czy_konczy_zdanie_updated_at Timestamp,
    sg_choice_id Integer Null default Null,
    sg_variant_id Integer Null default Null,
    dodany Boolean default 'f' not Null,
    chosen Boolean default 't', -- dedukowalne z sg_variant_anot
    chosen_updated_at Timestamp,
    superancja Text, -- zserializowana tablica [⟨poz_sym⟩* ]
    created_at Timestamp,
    updated_at Timestamp
);

create table transza(

```

```

    transza_id Integer primary key autoincrement,
    uzytkownik_id Integer,
    created_at TimeStamp,
    updated_at TimeStamp);

create table word_senses_rozbieznosc(
    word_senses_rozbieznosc_id Integer primary key autoincrement,
    token_id Integer not Null,
    sensy_id Integer not Null,
    created_at TimeStamp,
    updated_at TimeStamp
);

create view interpretacja_export as
select interpretacja_id as interpretacja_export_id,
interpretacja.* ,
lemat, klasa_gram_ozn
from interpretacja inner join leksem using( leksem_id )
inner join klasa_gram using( klasa_gram_id )
where interpretacja.dodana='f' or interpretacja.disamb='t'
order by numer_lex_token, interpretacja_id;

create index akapit_morph_id
on akapit( morph_id );

create index akapit_path_id
on akapit( path_id );

create unique index akapit_path_id_morphosyntactic_xmlid
on akapit( path_id, morphosyntactic_xmlid );

create unique index akapit_path_id_segmentation_xmlid
on akapit( path_id, segmentation_xmlid );

create index akapit_transzy_akapit_id
    on akapit_transzy( akapit_id );

create index akapit_transzy_odrzucony
    on akapit_transzy( odrzucony );

create index akapit_transzy_status
    on akapit_transzy( status );

create index akapit_transzy_transza_id
    on akapit_transzy( transza_id );

create index akapit_transzy_uzytkownik_id
    on akapit_transzy( uzytkownik_id );

create unique index cz_m_leksem_lemat_cz_m_id on cz_m_leksem( lemat,
cz_m_id);

create unique index cz_m_leksem_xmlid on cz_m_leksem( xmlid);

create index "index_session_on_session_id" on session( "session_id"
);

create index "index_session_on_updated_at" on session( "updated_at" );

```

```
create index interpretacja_anot_interpretacja_id
    on interpretacja_anot( interpretacja_id );

create index interpretacja_anot_token_id
    on interpretacja_anot (token_id);

create unique index interpretacja_anot_token_id_akapit_transzy_id
    on interpretacja_anot (token_id, akapit_transzy_id);

create index interpretacja_anot_token_id_uzytkownik_id
    on interpretacja_anot (token_id, uzytkownik_id);

create index interpretacja_anot_uzytkownik_id
    on interpretacja_anot (uzytkownik_id);

create index interpretacja_created_at on interpretacja( created_at);

create index interpretacja_cz_m_leksem_id on interpretacja(
cz_m_leksem_id);

create index interpretacja_cz_m_leksem_przypisany on interpretacja(
cz_m_leksem_przypisany);

create index interpretacja_disamb
on interpretacja( disamb );

create index interpretacja_leksem_id
on interpretacja( leksem_id );

create unique index interpretacja_path_id_msdxmlid
on interpretacja( path_id, msd_xmlid );

create index interpretacja_token_id
on interpretacja( token_id );

create index interpretacja_token_id_disamb
on interpretacja( token_id, disamb );

create unique index klasa_gram_klasa_gram_ozn
on klasa_gram (klasa_gram_ozn);

create index komentarz_akapit_id
on komentarz(akapit_id);

create index komentarz_akapit_transzy_id
on komentarz(akapit_transzy_id);

create index komentarz_created_at
on komentarz(created_at);

create index komentarz_uzytkownik_id_akapit_id
on komentarz(uzytkownik_id, akapit_id);

create index koniec_zdania_anot_akapit_id
on koniec_zdania_anot( akapit_id );

create unique index koniec_zdania_anot_akapit_transzy_id_token_id
on koniec_zdania_anot( akapit_transzy_id, token_id );

create index leksem_created_at on leksem( created_at);
```

```

create index leksem_cz_m_leksem_id on leksem( cz_m_leksem_id);

create index leksem_cz_m_leksem_przypisany on leksem(
cz_m_leksem_przypisany);

create index leksem_cz_m_przypisana on leksem( cz_m_przypisana);

create index leksem_lemat_klasa_gram_id
on leksem (lemat, klasa_gram_id);

create unique index leksem_path_id_lemat_klasa_gram_id
on leksem( path_id, lemat, klasa_gram_id );

create unique index leksem_path_id_lex_xmlid
on leksem( path_id, lex_xmlid );

create unique index morph_xmlid_path_text
on morph( xmlid, path_text );

create index morphosyntactic_rozbieznosc_token_id
      on morphosyntactic_rozbieznosc( token_id );

create index nowa_segmentacja_akapit_transzy_id on nowa_segmentacja(
akapit_transzy_id);

create unique index nowa_segmentacja_seg_is_seg_should_be_ids_nps
on nowa_segmentacja( seg_is, seg_should_be, ids, nps );

create index nowa_segmentacja_transza_id on nowa_segmentacja(
transza_id);

create unique index path_path_text on path( path_text );

create unique index poziomy_annotacji_uzytkownik_id
on poziomy_annotacji( uzytkownik_id );

create unique index prosby_annotatorek_dotyczy_type_dotyczy_id
on prosby_annotatorek( dotyczy_type, dotyczy_id);

create index protokol_akapit_id
      on protokol( akapit_id );

create index protokol_created_at
on protokol( created_at );

create index protokol_czy_rozbieznosc
      on protokol( czy_rozbieznosc );

create index protokol_poziom
      on protokol( poziom );

create index punkt_protokolu_akapit_transzy_id
      on punkt_protokolu( akapit_transzy_id );

create index punkt_protokolu_elt_protokol_id_elt_protokol_type
      on punkt_protokolu( elt_protokol_id, elt_protokol_type );

create index punkt_protokolu_protokol_id
      on punkt_protokolu( protokol_id );

create index punkt_protokolu_uzytkownik_id

```



```
    on punkt_protokolu( uzytkownik_id );

create index segmentation_rozbieznosc_sg_choice_id
    on segmentation_rozbieznosc( sg_choice_id );

create index sens_anot_akapit_transzy_id
on sens_anot( akapit_transzy_id );

create unique index sens_anot_akapit_transzy_id_token_id on
sens_anot( akapit_transzy_id, token_id);

create index sens_anot_interpretacja_id
on sens_anot( interpretacja_id );

create index sens_anot_interpretacja_id_akapit_transzy_id
on sens_anot( interpretacja_id, akapit_transzy_id );

create index sens_anot_sensy_id
on sens_anot( sensy_id );

create index sens_anot_token_id
on sens_anot( token_id );

create index sensy_cz_m_id on sensy( cz_m_id);

create unique index sensy_xmlid on sensy( xmlid);

create index sentences_rozbieznosc_token_id
on sentences_rozbieznosc( token_id );

create index sg_choice_akapit_id
on sg_choice( akapit_id );

create unique index sg_variant_anot_akapit_transzy_id_sg_variant_id
on sg_variant_anot( akapit_transzy_id, sg_variant_id);

create index sg_variant_sg_choice_id
on sg_variant( sg_choice_id );

create index statusy_akapit_id on statusy( akapit_id);

create unique index statusy_akapit_transzy_id
on statusy( akapit_transzy_id );

create index statusy_morphosyntactic on statusy(morphosyntactic);

create index statusy_named_entities on statusy(named_entities);

create index statusy_segmentation on statusy(segmentation);

create index statusy_sentences on statusy(sentences);

create index statusy_syntactic on statusy(syntactic);

create index statusy_syntactic_words on statusy(syntactic_words);

create index statusy_transza_id on statusy( transza_id);

create index statusy_word_senses on statusy(word_senses);

create index tagset_typ_lewe
on tagset(typ, lewe);
```

```

create index token_akapit_id
on token( akapit_id );

create index token_akapit_id_orth on token( akapit_id, orth);

create index token_akapit_id_segmentation_xmlid_orth
on token( akapit_id, segmentation_xmlid, orth );

create index token_czy_interp
on token( czy_interp );

create index token_dodany on token( dodany);

create unique index token_kolejnosc
on token( kolejnosc );

create index token_ns_nastepuje
on token( ns_nastepuje );

create index token_ns_poprzedza
on token( ns_poprzedza );

create index token_orth on token( orth);

create index token_path_id_morphosyntactic_xmlid
on token( path_id, morphosyntactic_xmlid );

create unique index token_path_id_segmentation_xmlid
on token( path_id, segmentation_xmlid );

create index token_sg_choice_id
on token( sg_choice_id );

create index token_sg_variant_id
on token( sg_variant_id );

create index transza_uzytkownik_id
      on transza (uzytkownik_id);

create index word_senses_rozbieznosc_token_id
      on word_senses_rozbieznosc( token_id );

create TRIGGER nowa_segmentacja_total_akapitow
  after insert on nowa_segmentacja
begin
  update nowa_segmentacja set total_akapitow = (select Count(* ) from
    akapit)
  where nowa_segmentacja_id = new.nowa_segmentacja_id;
end;

create TRIGGER token_default_kolejnosc
  after insert on token
when (select kolejnosc from token where token_id = new.token_id) is
  Null
begin
  update token set kolejnosc = (new.token_id * 216)
  where token_id = new.token_id;
end;

```

## The Ruby and RHTML code

### app/controllers

#### admin\_controller.rb

```
23   require 'interpretacja_anot'
24
25   class AdminController < ApplicationController
26
27     layout "admin"
28
29     before_filter :authorize
30     before_filter :update_activity_time, :except => :session_expiry
31     before_filter :check_admin, :except => :zmien_haslo
32
33     # ssl_required :dodaj_uzytkownika, :usun_uzytkownika, :zmien_haslo,
34     :lista_uzytkownikow, :zablokuj_uzytkownika
35
36     def dodaj_uzytkownika
37       @uzytkownik = Uzytkownik.new(params[:uzytkownik])
38       if request.post? and @uzytkownik.save
39         flash.now[:notice] = "Stworzyłam użytkownika
40         #{@uzytkownik.login} (#{@uzytkownik.rola.opis_kr})."
41         @uzytkownik = Uzytkownik.new
42       end
43     end
44
45     def usun_uzytkownika
46       if request.post?
47         uzytkownik = Uzytkownik.find(params[:id])
48         if Rola.usuwalny?( uzytkownik )
49           if Rola.anotator?( uzytkownik.rola_id ) # dołożone na życzenie
50             AP 2009/6/5
51             # anotatorowi odbieramy wszystkie transze
52             uzytkownik.transze_otwarte.each{ |trao|
53               traو.uzytownik_id = nil
54               traو.save! }
55             end
56             uzytkownik.destroy
57             flash[:notice]="Usunęłam użytkownika #{@uzytkownik.login}"
58             redirect_to(:action => :lista_uzytkownikow)
59           else
60             flash[:notice] = "Nie usuniesz takiego użytkownika, ewentualnie
61             możesz go zablokować"
62             redirect_to (:action => "zablokuj_uzytkownika", :id =>
63             uzytkownik)
```

```

61         end
62     end
63 end # of usun_uzytkownika

66 def zablokuj_uzytkownika
67     if request.post?
68         uzytkownik= Uzytkownik.find(params[:id])
69         if uzytkownik.uzytkownik_id != session[:uzid]
70             moznablokowac=true
71         else
72             moznablokowac=false
73             flash[:notice] = 'Nie będziesz blokować samej/go siebie'
74         end
75         if moznablokowac
76             if false and Rola.anoator?( uzytkownik.rola_id ) # wyłączone
na życzenie AP 2009/6/5
77                 # anotorowi odbieramy wszystkie transze
78                 uzytkownik.transze_otwarte.each{ |trao|
79                     traو.uzytkownik_id = nil
80                     traو.save! }
81             end
82             uzytkownik.rola_id = Rola.rid(:zablokowany)
83             uzytkownik.save!
84             flash[:notice]= "Zablokowałam użytkownika #{uzytkownik.login}"
85         end
86         redirect_to( :action => :lista_uzytkownikow )
87     end
88 end# of zablokuj_uzytkownika

91 def lista_uzytkownikow
92     @wszyscy_uzytkownicy = Uzytkownik.find(:all, :order => :login)
93 end

96 def zmien_haslo
97     @uzytkownik = Uzytkownik.find(params[:id])

99     if (@uzytkownik.uzytkownik_id != session[:uzid]) and
100     ( session[:rola_id] & Rola.rid(:admin) ==0)
101         flash[:notice] = "Nie masz uprawnień do wykonania tej czynności.
Za karę wylogowuję Cię."
102         redirect_to(:controller => "logowanie" , :action =>
"wylogowanie" )

104     elsif request.post? and params[:uzytkownik]
105         stare_haslo = params[:uzytkownik][:stare_haslo]

107         if session[:rola_id] & Rola.rid(:admin) != 0 or
108         ( @uzytkownik.uzytkownik_id == session[:uzid] and
109         stare_haslo and
110         @uzytkownik.hasz_haslo ==
111         Uzytkownik.haslo_szyfrowane(stare_haslo, @uzytkownik.sol))

113         @uzytkownik.haslo=params[:uzytkownik][:haslo]

```

```

114      @uzytkownik.haslo_confirmation=params[:uzytkownik][:haslo_confirmation]
115      @uzytkownik.haslo_updated_at = Time.now
116      if @uzytkownik.save
117        if session[:rola_id] & Rola.rid(:admin) != 0
118          flash[:notice] = "Zmieniłam hasło użytkownika
119            #{@uzytkownik.login}."
120          redirect_to(:action => :lista_uzytkownikow)
121        else
122          flash[:notice] = "#{@uzytkownik.login}, właśnie
123            zmieniłeś/aś swoje hasło."
124          redirect_to(:controller => :logowanie, :action => :index)
125        end
126      end
127    else
128      if session[:rola_id] & Rola.rid(:admin) != 0
129        redirect_to(:action => :lista_uzytkownikow)
130      else
131        flash[:notice] = "Zmiana hasła nie powiodła się (złe stare
132          hasło)"
133        redirect_to(:action => :zmien_haslo)
134      end
135    end
136  end
137 end # of zmien_haslo
138 end # of class

```

### **anotacja\_controller.rb**

```

23  Uzytkownik.find( :first ) # żeby utworzyć Natrorów, gdyby ich nie było.
24
25  class AnotacjaController < ApplicationController
26
27    include ApplicationHelper
28
29    layout "admin"
30
31    GOSCIENNE = # mogą oglądać wszyscy
32    [ :ogladaTwo, :ustal_akapit, :anotuj, :pokaz_transze,
33      :wyszukaj_akapity,
34      :pokaz_quasi_transze, :ukryj_quasi_transze,
35      :ukryj_wszystkie_quasi_transze,
36      :quasi_transza_brutal,
37      :sense_long_toggle,
38      :sense_inventory,
39      :sense_long_hide_all,
40      :zweryf,
41      :wsze_transze_otwarte_toggle,
42      :wsze_transze_BW_toggle
43    ]
44
45    ANNOTABLES_ET_AUDITABLES = [ :komentarz_dodaj, :komentarz_anulowany,
46      :komentarz_zatwierdz,

```

```

47             :nowa_segmentacja_ukryj,
48             :nowa_segmentacja_pokaz,
49             :nowa_prosba_segmentacyjna
50         ]
51
52     ANNOTER_ONLY = [:pobierz_transze, :prosba_o_odebranie_transzy,
53                   :prosba_anulowana, :prosbe_o_odebranie_zapisz,
54                   :odebrania_transz_ukryj, :odebrania_transz_pokaz,
55                   :nie_chce_transzy, :ramka_prosba_o_wzorzec]
56
57     WSEWID_ONLY = [:spełnij_prosbe, :odrzuć_prosbe ]
58
59     before_filter :authorize
60     before_filter :przetwarzam_nie_przeszkadzac
61     after_filter :skonczyłam_przetwarzac
62
63     before_filter :update_activity_time, :except => :session_expiry
64
65     before_filter :check_annotant, :except => GOSCINNE
66     + ANNOTABLES_ET_AUDITABLES +
67     ANNOTER_ONLY + WSEWID_ONLY
68     # check_annotant patrzy czy (anotator lub zarządca)
69     before_filter :check_voyeur, :only => GOSCINNE
70     before_filter :check_annotant_ou_auditeur, :only =>
71     ANNOTABLES_ET_AUDITABLES
72     before_filter :check_anotator, :only => ANNOTER_ONLY
73     before_filter :check_wsewid, :only => WSEWID_ONLY
74
75     # check_annotant_ou_auditeur patrzy czy (anotator lub zarządca lub
76     audytor)
77     ## before_filter :check_zaradca, :only => ZARZADCA_ONLY
78     # before_filter :check_idid
79
80     EDYTOWALNY_EXCEPTIONS = [
81         :lista_transz, :transze_zamkniete_pokaz,
82         :transze_zamkniete_ukryj,
83         :dyszamb_anulowana,
84         :sensowanie_anulowane,
85         :fraz_wybor_anulowany,
86         :segmentacja_anulowana,
87         :poziomuj
88     ] +
89     GOSCINNE + ANNOTABLES_ET_AUDITABLES +
90     ANNOTER_ONLY + WSEWID_ONLY
91     before_filter :check_edytowalny, :except => EDYTOWALNY_EXCEPTIONS
92
93     before_filter :check_drag_queen, :only => :pobierz_transze # reszta
94     sprawdza się w check_edytowalny.
95
96     def pobierz_transze
97         @anotator = Uzytkownik.find( session[:uzid] )
98         if r_anotator?
99             if @anotator.transze_otwarte( :refresh ).size <
100             Transza::MAX_TRANSZ

```

```

99     transze_zabronione = @anotator.odebrania_odebrane.collect {
100       |ode|
101       ode.dotyczy_id # to jest transza_id, tylko w powiązaniu
102       polimorficznym.
103     } + @anotator.transza.collect {
104       |tra| traid=tra.transza_id; ((1+(traid-1)/4*
105       4)..((traid+3)/4* 4)).to_a}.flatten
106     tr_przypisywana = Transza.find( :first,
107                                     :conditions =>
108                                     " transza_id = ( select
109                                     min(transza_id) as minid" +
110                                     " from transza ta where
111                                     ta.transza_id not in " +
112                                     " (#{transze_zabronione.join(',
113                                     ')})) " +
114                                     " and uzytkownik_id is null ) ")
115     if (not tr_przypisywana)
116       flash[:notice]="Brak transz do przypisania"
117     else
118       ##          flash[:notice]="Pierwsza dopuszczalna to
119       #{pierwsza_tr_dopuszczalna}"
120       tr_przypisywana.uzytkownik_id=@anotator.uzytkownik_id
121       tr_przypisywana.save!
122     end
123     render :partial => 'lista_transz', :object =>
124     @anotator.transze_otwarte( :refresh )
125     ##          redirect_to(:action => :lista_transz)
126   else # to nie anotator
127     flash[:notice] = "Próbowaliśmy/eś czynności dostępnej tylko
128     anotatorom!"
129     redirect_to( :controller => :logowanie, :action => :wylogowanie )
130   end
131 end # of pobierz_transze

132
133 def lista_transz
134   session[:anotator] = nil if r_anotator?
135   @anotator = Uzytkownik.find(session[:uzid])
136   @sa_transze_zamkniete = ( @anotator.transze_zamkniete.size > 0 )
137   @odebrania_transz = @anotator.odebrania_transz
138 end

139
140 # Na życzenie Adama Przepiórkowskiego
141 # w jego wersji ukrywamy prośbę o odebranie transzy na osobnej stronie.
142 def nie_chce_transzy
143   session[:anotator] = nil if r_anotator?
144   @anotator = Uzytkownik.find(session[:uzid])
145   @odebrania_transz = @anotator.odebrania_transz
146 end

147
148 def transze_zamkniete_pokaz

```

```

145     session[:transze_zamkniete_pokaz] = true
146     redirect_to :action => :lista_transz
147 end
148
149 def transze_zamkniete_ukryj
150     session[:transze_zamkniete_pokaz] = false
151     redirect_to :action => :lista_transz
152 end
153
154 def pokaz_transze
155     conditiones = { :transza_id => ( params[:id] || session[:transza]
156     ) }
157     # anotorowi pokażemy transzę tylko, jeśli jest jego
158     if r_anotator?
159         conditiones[:uzytkownik_id] = session[:uzid]
160     end
161     @transza=Transza.find( :first, :conditions => conditiones)
162     if @transza
163         @akapity=(unless r_gosc? :
164             @transza.akapity_transzy_rozbiezne(:refresh) +
165             @transza.akapity_transzy_niezatwierdzone(:refresh) +
166             @transza.akapity_transzy_zatwierdzone(:refresh)
167             else [] end +
168             @transza.akapity_transzy_zamkniete(:refresh)).collect
169             {|akat|
170                 [akat.akapit, akat] }
171         session[:transza]=@transza.transza_id
172         zniluj_akapitowe( :weryfikacja )
173         session[:anotator] = @transza.uzytkownik_id unless r_anotator?
174         session[:anotator] = nil if r_anotator?
175     else # gdy transza is nil
176         if r_anotator?
177             flash[:notice] = "Prawdopodobnie nie masz uprawnień do
178             oglądania wybranego akapitu: nie znalazłam transzy
179             #{params[:id]} wśród Twoich."
180             redirect_to( :action => :lista_transz )
181         else redirect_to( :action => :ogladactwo )
182         end
183     end
184 end # of pokaz_transze.
185
186 def wyszukaj_akapity
187     zp = 'zawęż poprzednie'
188     nowe_zapytanie=true
189     zawez_poprzednie=false
190     sa_kryteria = false
191     if request.post? and params[:szukaj]
192         # jeśli to post, to spodziewamy się danych z formularza (parametry
193         zapytania)
194         klucze = [:szukaj_lemat, :szukaj_sens, :szukaj_klasa]
195         klucze.each{ |klucz|

```



```

193         pk = params[:szukaj][klucz]
194         pk = nil if pk == ""
195         sa_kryteria = true if pk
196         session[klucz] = pk}
197         if session[:szukaj_commit] == zp and params[:commit] != zp
198             nowe_zapytanie = true
199         end
200         zawez_poprzednie = true if params[:commit] == zp
201         session[:szukaj_commit] = params[:commit]
202     end# of request.post? etc.
203     if sa_kryteria
204         if nowe_zapytanie
205             akats = AkapitTranszy.wyszukaj(
206                 session[:szukaj_lemat],
207                 session[:szukaj_sens],
208                 session[:szukaj_klasa])
209         end
210
211         if akats and akats[0]
212             sakat = session[:szukaj_akapity_transzy]
213             akats_poprz = sakat if zawez_poprzednie and sakat
214             session[:szukaj_akapity_transzy] = akats.collect{ |akat|
215                 akat.akapit_transzy_id }
216             session[:szukaj_akapity_transzy] &= akats_poprz if
217                 zawez_poprzednie and akats_poprz
218         elsif akats == [] : session[:szukaj_akapity_transzy] = nil
219         end
220
221         if session[:szukaj_akapity_transzy] and ((not akats) or
222             zawez_poprzednie)
223             akats = AkapitTranszy.find( session[:szukaj_akapity_transzy] )
224         end
225
226         @akapity = akats.collect{|akat| [akat.akapit, akat] } if akats
227         zniluj_akapitowe( :weryfikacja )
228     else
229         flash[:notice] = "Podaj choć jedno kryterium wyszukiwania!"
230     end
231 end # of wyszukaj_akapity.
232
233 def ustal_akapit # wołany z widoku pokaz_transze.rhtml,
234     wyszukaj_akapity.rhtml, i ogladactwo.rhtml.
235     zniluj_akapitowe( :weryfikacja )
236     ## logger.info '@@@ params: ' + params.inspect
237     session[:akapit] = params[:akapit].to_i if params[:akapit]
238
239     if params[:goto] and params[:goto][:akapit]
240         blubra = params[:goto][:akapit].tr('^0123456789', "").to_i
241         session[:akapit] = blubra if blubra > 0
242     # if we get the akapit id from a form in ogladactwo view, we trim it of
243     # anything non decimal.
244     end

```

```

244 session[ :anotator ] = params[ :anotator ].to_i if params[
    :anotator ]

246 session[ :akat ] = params[ :akat ].to_i if params[ :akat ]

248 if params[ :goto ] and params[ :goto ][ :akat ]
249   blubra = params[ :goto ][ :akat ].tr('^0123456789', '').to_i
250   session[ :akat ] = blubra if blubra > 0
251 end

253 if session[ :akat ]
254   session[ :transza ] =
255     AkapitTranszy.find( session[ :akat ] ).transza_id

257 elsif session[ :anotator ] and session[ :akapit ]
258   akat = AkapitTranszy.znajdz(session[ :akapit ], session[
    :anotator ])
259   session[ :akat ] = akat.id
260   session[ :transza ] = akat.transza_id

262   # poniższy elsif dołożyłem 2010/4/9
263   elsif Rola.wszewid?( session[ :rola_id ] ) and session[ :akapit ]
264     akat = AkapitTranszy.find_by_akapit_id( session[ :akapit ] )
265     session[ :akat ] = akat.id
266     session[ :transza ] = akat.transza_id
267   else
268     raise
269     ###{annotation_controller.capitalize}Controller.ustal_akapit:
        parameters don't suffice to determine akapit_transzy."
270   end

271   ##      logger.info "==== session[ :jak_anotator ]=#{session[
    :jak_anotator ].inspect}"

273   redirect_to :action => :anotuj, :akat => session[ :akat ], :anchor
    => :primafalsa
274 end # of ustal_akapit

278 def anotuj
279   ##                                     logger.info "==== aca w. 215.
    #{Time.now}"
281   ##      logger.info "==== session[:weryfikacja] =
    #{session[:weryfikacja].inspect}"

284   set_uzid
285   @komentarz_do_odrzucenia = true if params[:kom_odrzuc]
286   inicjuj_akat_i_poziomy
287   zrob_ramke_dup # tam jest if
288   ##      @transza = ( session[ :transza ] && Transza.find(
    session[:transza] ) ) # poprawione na koniunkcję 2009/8/22.
289   # sprawdź, czy akapit z jakiejś jego transzy

291   logger.info "==== #316 ak.t. #{@akapit_transzy.id} - statuski:
    #{@akapit_transzy.statuski}"
292   @akapit_transzy.autozatwierdź_word_senses if @akapit_transzy

```

```

294     logger.info "### #316 ak.t. #{@akapit_transzy.id} - statuski:
        #{@akapit_transzy.statuski}"

296     if (@transza or r_wszewid? # wszechwidzący nie zawsze pracują
        w transzach.
297         ) and @akapit_transzy and
298         ( @akapit_transzy.uzytownik_id == @anotid or
299         @akapit_transzy.get_uzid == @anotid or r_wszewid?
        # wszechwidzący może chcieć oglądać akapit jeszcze nie
        przydzielony, jeśli jest on bliźniaczem jakiegoś przydzielonego.
300         )
301     @akapit = @akapit_transzy.akapit
302     session[ :status ] = @akapit_transzy.status
303     ##                                     logger.info "### aca w. 229.
        #{Time.now}"
305     if session[ :weryfikacja ] and session[:weryfikacja][:werdykt]
        == -1
306         @weryfikacja = session[:weryfikacja]
307         # w tym wypadku mamy primafalsa, do której będziemy kotwiczyć,
        więc nie kotwiczymy sensu.
308     else
309         verifiabes = @poziomy.verifiabes( @akapit_transzy )
310         ##         logger.info "### verifiabes: " + verifiabes.inspect
311         if verifiabes[ 0 ]
312             # weryfikujemy, i to do końca, żeby ewentualnie mieć pełną listę
            niezgodności
313             @weryfikacja = @akapit_transzy.weryfikacja(
314                 :dokonca => true,
315                 :rola_id =>
316                 session[:rola_id],
317                 :poziomy => verifiabes,
318                 :właśnie_zatwierdzamy =>
319                 false,
320                 :session_uzid => session[
321                 :uzid ]
322             )
323             session[:weryfikacja] = @weryfikacja
324             end # of if verifiabes

324         ##         logger.info "### " + @weryfikacja.inspect
325     end # of if session[:weryfikacja] or not

328     # następnych 6 wierszy było dotąd powyżej powyższego end,
329     # co powodowało #252
330     @werdykt = ( @weryfikacja ||= { :werdykt => 0 } )[:werdykt]
331     if ( not @weryfikacja[ :primafalsa ] ) and
332         @poziomy.word_senses?
333         # jeśli mamy wyświetlać do anotacji sensy słów, to ustawimy
        kotwiczkę na pierwszy sens
334         @akapit_transzy.akapit.set_primafalsa(

```

```

335                                     @weryfikacja,
                                     @akapit_transzy.primus_sensibilis
                                     )
336     end # of if not prima falsa.
338     logger.info "### #316 ak.t. #{@akapit_transzy.id} - statuski:
        #{@akapit_transzy.statuski}"

341     # pierwotnie @bluzid był ustawiany tylko gdy werdykt ==-1,
342     # ale może go potrzebować zarządca do zmiany strony
343     @bluzid = @akat_bliźniak.get_uzid( session[ :uzid ] )
344     @bliakath = @akat_bliźniak.akat_hash
345     @statuski = @akapit_transzy.statuski
346     @do_poprawki = ( Rola.anotator?( session[:rola_id] ) and
347                     ( @statuski.include?( AkapitTranszy::STATI[
348                         :do_poprawki] ) or
349                     @statuski.include?( AkapitTranszy::STATI[
350                         :po_poprawce] ) )
351                     )
352     @pokaż_bliż = ( @akapit_transzy.ma_Status?( :>=, :do_osądenia )
353     or
354     Rola.wszewid?( session[:rola_id] ) )
355 else # nie ma transzy lub akapit_transzy lub @transza_uzytkownik_id
356 <> ...
357     flash[:notice] = "Niewłaściwa deskrypcja akapitu: "+
358         "akat_id: #{session[:akat]} akapit_id: #{session[:akapit]};
359         anotator: #{session[:anotator]}; transza_id:#{session[
360             :transza]}."
361     # Jeśli ta notycja się wyświetli, to znaczy, że transza była dobrze,
362     # bo gdyby źle, to by się pojawiła notycja z przekierowania nie
363     # znalezienia transzy.
364     redirect_to( :action => :pokaz_transze, :id => session[:transza]
365     )
366 end
367 ##     logger.info "### aca w. 265. #{Time.now}"
368 end # of anotuj.

369 def pokaz_interpretacje
370     render :partial => 'token_wszi', :object => [akapit_transzy,
371     Token.find(params[:id])]
372 end

373 def nowa_dyzambiguacja
374     set_uzid
375     intid=params[:interpretacja_anot]
376     if intid
377         intid=intid[:interpretacja_id].to_i
378         tok = Interpretacja.find( intid ).token
379         @tokid = tok.id
380         @interpretacja_anot=InterpretacjaAnot.find(
381             :first,
382             :conditions => {

```

```

379                                     :akapit_transzy_id =>
                                     akapit_transzy.id,
380                                     :token_id => @tokid
381                                 })
382     @interpretacja_anot ||= InterpretacjaAnot.new
383     @interpretacja_anot.interpretacja_id = intid
384     @interpretacja_anot.uzytownik_id=@uzid
385     @interpretacja_anot.akapit_transzy_id = akapit_transzy.id
386     @interpretacja_anot.token_id = @tokid
387     InterpretacjaAnot.transaction do
388       @interpretacja_anot.save!
389       if DlaEli.nk
390         obniz_status( akapit_transzy, :morphosyntactic)
391         tok.odsmiec_interpretacje
392       else
393         obniz_status( akapit_transzy )
394       end
395     end
396     else# not intid
397       @tokid=params[:id].to_i
398     end
399     ##      redirect_to :action => 'anotuj' # for debug
400     raiseifzero( @tokid, "nowa_dyzambiguacja" )
401     if ( szd = session[ :zmieniaj_dyzamb ] ) and szd[ @tokid ]
402       szd[ @tokid ] = nil
403     end
404     odswiez_token(@tokid)
405     tok.zambiguuj
406     end # of nowa_dyzambiguacja.

410     def dyzamb_anulowana
411     # to by mogło iść do dyzamb_anulowana.rjs, ale nie chce mi się mnożyć
    plików
412       @tokid=params[:id].to_i
413       szd = session[ :zmieniaj_dyzamb ]
414       if szd and szd[ @tokid ]
415         szd[ @tokid ] = nil
416       end
417       raiseifzero( @tokid, "dyzamb_anulowana" )
418       odswiez_token(@tokid)
419     end

422     def pokaz_sensy
423       set_uzid
424       akat_hash
425       tok187 = Token.find(params[:id])
426       @the_sens = tok187.the_sens( @akat_hash )
427       render :partial => 'token_wybs', :object =>
428         [ @akapit_transzy,
429           tok187,
430           tok187.do_sensu?( @akat_hash ) ]

```

```

431 end # of pokaz_sensy

434 def wybor_sensu # wołana z _token_wybs.rhtml
435   set_uzid
436   akat_hash # inicjuje @akat_hash, jeśli jej nie było
437   obniz_status( akapit_transzy, :word_senses )
438   sensid = params[ :the_sens ]
439   tokid = params[ :id ].to_i
440   disaid = params[ :disamb_id ].to_i
441   raiseifzero( tokid, "wybor_sensu" )

443   if sensid and sensid[:sensy_id] and
444     sensid[:sensy_id].to_i != 0
445     # drugi człon koniunkcji na wypadek,
446     # gdyby nic nie wybrano, ale to i tak mało
447     sensid=sensid[ :sensy_id ].to_i

449     @sens_annot=SensAnot.znajdz( tokid, @akat_hash )

451     @sens_annot ||= SensAnot.new

453     @sens_annot.sensy_id = sensid
454     @sens_annot.uzytownik_id = @uzid
455     @sens_annot.akapit_transzy_id = akapit_transzy.id
456     @sens_annot.interpretacja_id = disaid
457     @sens_annot.token_id = tokid
458     @sens_annot.automatycznie = false
459     s1187 = @sens_annot.sensy( :refresh )
460     if s1187
461       @sens_annot.save!
462     else
463       @sens_annot = nil
464     end
465   end # of sensid and sensid[:sensy_id] and sensid[:sensy_id].to_i != 0
466   odswiez_token(tokid, :tylko_sens)
467 end # of wybor_sensu

470 def sensowanie_anulowane
471   t187 = Token.find( params[:id] )
472   ## @the_sens = t187.the_sens( akat_hash )
473   ## render :partial => 'token_sens', :object => [t187,
474     @akapit_transzy, true]
475   odswiez_token( t187.id, :tylko_sens )
476 end

480 def sense_long_toggle
481   akapit_transzy
482   eltid192 = params[ :id ]
483   ids192 = eltid192.split( '_' )[1, 3].collect { |i192| i192.to_i }

485   session[ :show_long_sense ] ||= []
486   if session[ :show_long_sense ].include?( eltid192 )
487     session[ :show_long_sense ].delete( eltid192 )
488   else

```

```

489     session[ :show_long_sense ] « eltid192
490   end
491
492   arr192 = [ ids192[0], ids192[1], Sensy.find( ids192[2] ) ]
493
494   render :update do |page|
495     page.replace eltid192,
496     :partial => 'sens_long',
497     :object => arr192
498
499     page.replace eltid192 + '-link',
500     :partial => 'sens_pelny_opis_link',
501     :object => arr192
502   end
503 end # of sense_long_toggle
504
505 def sense_long_hide_all
506   # używana w widoku sense_inventory
507
508   session[ :show_long_sense ] ||= []
509   session[ :show_long_sense ].delete_if { |eltid192|
510     eltid192 =~ /666_6666\d\d\d\d\d/ }
511   redirect_to :action => :sense_inventory
512 end
513
514 def zatwierdz_akapit
515   # to tylko wykonuje zatwierdzenie akapitu przez Anotatora. Weryfikacji
516   # i ewentualnego dokonania zmian w rekordach dokona
517   # zweryfikuj_akapit i jej metody commit_....
518   akat = akapit_transzy
519   if session[:zdziew_sie_akapitowi] : zdziew_sie = nil
520   else zdziew_sie = :zdziew_sie
521   end
522
523   poziomy = params[:poziom].split('/').collect{ |poz| poz.intern }
524   # sprawdź każdy poziom anotacji, którego dotyczy żądanie zatwierdzenia
525   poziomy.freeze
526
527   niezatw_notice = "
528
529   # superanotatorowi sprawdzimy, czy rozstrzygnął wszystkie rozbieżności:
530   if r_zarzadca? and w = session[ :weryfikacja ]
531     rozb =
532     poziomy.collect { |poz|
533       if w[poz] and w[poz][0] # to mamy tablicę id-ów tokenów, oby
534       pustą
535         "na poziomie #{PoziomyAnotacji.gen(poz)} w tokenie/nach " +
536         w[poz].collect{ |id| "»" + Token.find( id).orth + "«"
537         }.join( ', ' )
538       end
539     }.compact.join( ';' )
540   if rozb != "
541     niezatw_notice = "Nie zatwierdzam: nie rozstrzygnąłeś
542     rozbieżności #{rozb}!"

```

```

540     end
541   end # of if zarządca (superanotator)

543   poziomy_niegotowe = []

545   poziomy.each{ |poz|
546     self.send( "zatw_#{poz}", zdziw_sie, poziomy_niegotowe,
547               niezatw_notice )
548   }

549   unless niezatw_notice[0] # niezatw_notice jest <stringiem>, pytamy
550     tutaj, czy niepustym.
551     logger.info "poziomy: #{poziomy.inspect}"
552     zweryfikuj_akapit(akat, :zatwierdz, poziomy)

553   else
554     session[:zdziw_sie_akapitowi] = :dziwie_sie unless /^nie/i =~
555       niezatw_notice
556     poziomy_niegotowe.each{ |poz|
557       obniz_status( akat, poz )
558     }
559     # ale mogą być poziomy gotowe, i te zweryfikujemy (błąd #157
560     2009/7/3)
561     poziomy_gotowe = poziomy - poziomy_niegotowe
562     if poziomy_gotowe[ 0 ]
563       zweryfikuj_akapit(akat, :zatwierdz, poziomy_gotowe,
564       niezatw_notice )
565     else
566       flash[:notice]= niezatw_notice
567       redirect_to :action => :anotuj, :akat => akat, :anchor =>
568       :notizia
569     end
570   end # of unless niezatw_notice.
571   ##      logger.info "##### session[:weryfikacja] =
572   ##      #{session[:weryfikacja].inspect}"
573 end # of zatwierdz_akapit.

574 # tu następują metody zatwierdzenia na poszczególnych poziomach

575   PoziomyAnotacji.poziomy_x.each{ |poz|
576     eval " def zatw_#{poz}( zdziw_sie, poziomy_niegotowe,
577       niezatw_notice )  logger.info \"### zatw_#{poz} \"  akat =
578       akapit_transzy  #{poz}_niegot = akat.#{poz}_niegot?( session[
579       :rola_id ], zdziw_sie )  if #{poz}_niegot[ 0 ]  niezatw_notice
580       « #{poz}_niegot  poziomy_niegotowe « :#{poz}  else
581       podnies_status(akat, :#{poz} )  logger.info \"###
582       podnies_status in zatw_#{poz}\"  AkapitTranszy.transaction do
583       akat.odrzucony=false  akat.save!  end # of transaction
584     end end "  # pytamy w tym warunku, czy niezatw_notice, które jest
585     Stringiem, ma pierwszy znak – czyli czy jest niepuste.

586     # of transaction.
587     # of if segm_niegot[0].
588     # of zatw_<poz>.

```



```

579     }

581     # powyższy kod tworzy nam dla każdego poziomu anotacji metodę
    zatw_<poziom>, mającą argumenty
582     # ( zdziw_sie, poziomy_niegotowe, niezatw_notice ).
583     # poziomy_niegotowe to <array>, do której dopiszemy ten poziom, jeśli
    nie jest gotowy,
584     # niezatw_notice to budowany w zatwierdzałkach kolejnych poziomów
    komunikat o niezatwierdzeniu
585     # powyższe dwa parametry będziemy modyfikować, czyli obsługiwać jako
    przekazane przez zmienną (wszystko w Rubym jest przekazywane przez
    zmienną, bo wszystko jest referencją do obiektu)
586     # zdziw_sie to symbol lub nil – mówi, czy mamy się tylko dziwić, czy
    zatwierdzać do końca mimo ewentualnego zdziwienia. Akurat
    w przypadku segmentacji nie ma się czemu dziwić.

587     #
588     # Metody te wołają akat.<poziom>niegot?( session[ :rola_id ],
    zdziw_sie ).

591     def odrzuc_akapit
592       raise "##{annotation_controller.capitalize}.odrzc_akapit: action
        forbidden."
593       akat = akapit_transzy
594       ko = akat.komentarz( session[:uzid], session[
        :tylko_nowe_komentarze ] )
595       if ko.size > 0 and ko[-1].przyczyna_odrzucenia != 0 and
596         ( /[a-a]/i =~ ko[-1].tresc.to_s or ko[-1].przyczyna != "" )

598       podnies_status( akat )
599       ## logger.info "@@@ podnies_status in odrzuc_akapit"
601       AkapitTranszy.transaction do
602         akat.odrzucony=true
603         akat.save!
604         if r_zarzacca?
605           zbli = akat.bliźniaczy
606           zbli.odrzucony = true
607           zbli.save!
608         end
609       end # of transaction

611       logger.info "poziomy: #{poziomy.join(', ')}"
612       zweryfikuj_akapit(akat, :odrzc, poziomy)

614     else # brak komentarza lub przyczyny odrzucenia lub p.o. jest "inne" i
    nie ma treści komentarza
615       notice = "Przy odrzuceniu musisz dodać komentarz i&nbsp;wybrać w
        nim przyczynę odrzucenia."
616       if ko[-1] and ko[-1].przyczyna_odrzucenia == 4
617         notice += " Podałaś/eś przyczynę Inne. Musisz napisać
        dokładniej, o&nbsp;co chodzi."
618       end
619       flash[:notice] = notice

```

```

620 # @komentarz_do_odrzucenia = true nie działa, tzn akcja anotuj tego nie
widzi.
621     redirect_to :action => :anotuj, :kom_odrzuc => true, :akat =>
    akat
622     end
624 end # of odrzuc_akapit.

627 def komentarz_dodaj
628     komentarz_toggle( true )
629 end

631 def komentarz_anulowany
632     komentarz_toggle( false )
633 end

636 def komentarz_zatwierdz
637     ko = Komentarz.new(:akapit_transzy_id => params[:id],
638                       :akapit_id => ( params[ :akapit_id ].to_i ||
        session[:akapit] ) ,
639     :uzytkownik_id => ( session[ :drag_queen_id ] ||
        session[ :uzid ] )
640     )
641     ko.tresc= params[:komentarz_nowy][:tresc]
642     ko.przyczyna_odrzucenia =
        params[:komentarz_nowy][:przyczyna_odrzucenia].to_i
643     if ( ko.tresc and /\w/i =~ ko.tresc ) or
644     ( ko.przyczyna_odrzucenia != 0 and ko.przyczyna_odrzucenia !=
        4)
645     ko.save!
646     end
647     render :update do |page|
648     page.replace_html 'komentarze',
649     :partial => 'komentarze',
650     :object => [ AkapitTranszy.find(params[:id]), :anotacja ]
651     page.replace_html 'komentarz_dodaj',
652     :partial => 'komentarz_dod_link',
653     :object => AkapitTranszy.find(params[:id])
654     end
655 end # of komentarz_zatwierdz

658 def fraza_upuszczono
659     tokpocz, tokkon = params[:id].to_i, params[:upusc].to_i
660     tokpocz, tokkon = tokkon, tokpocz if tokpocz > tokkon
661     akat = akapit_transzy

663 # sprawdzamy, czy nie próbujemy objąć już jakiejś zaznaczonej frazy
664     drogawolna=true
665     tokpocz.upto(tokkon) { |tokid0|
666     drogawolna = false if akat.fraza(tokid0)
667     }

669     if drogawolna

```

```

670     AkapitTranszy.transaction do
671       obniz_status( akat, :syntactic )
672       akat.fraza_anot.create(
673         :token_id_pocz => tokpocz,
674         :token_id_kon => tokkon)
675     end
676   end

678   refresh_fraza(tokpocz, tokkon)
679 end # of fraza_upuszczono

681 def frazy_wyczysc
682   akat = akapit_transzy
683   AkapitTranszy.transaction do
684     obniz_status( akat, :syntactic )
685     akat.fraza_anot.delete_all
686   end

688   redirect_to :action => :anotuj, :akat => akat, :anchor =>
:primafalsa
689 end # of frazy_wyczysc

691 def fraza_wyczysc
692   FrazaAnot.transaction do
693     FrazaAnot.delete(params[:id])
694     obniz_status( akapit_transzy, :syntactic )
695   end
696   redirect_to :action => :anotuj, :akat => akapit_transzy, :anchor
=> :primafalsa # najprościej
697   # zrenderować całą stronę, żeby odbudować brakujące komórki
698 end

700 def fraza_ukryj_typ
701   fraa=   FrazaAnot.find(params[:id])
702   @pokaz_wybor_typu = true
703   refresh_fraza( fraa.token_id_pocz, 0 )
704 end

706 def fraza_wybor_typu
707   set_uzid
708   akat_hash
709   fraa=   FrazaAnot.find(params[:id])
710   ftypid = params[:fraza_typ][:fraza_typ_id]
711   if ftypid
712     ftyp = FrazaTyp.find(ftypid)
713     vp = (ftyp.typ_symbol == "VP")
714     vpneg = nil
715     if vp
716       fraa.tokens.each{ |tok|
717         vpneg = true if tok.orth =~ /^nie$/i
718       }
719     end

721     synh = fraa.synheads(@akat_hash, ftypid)[0]

```

```

722     semh = fraa.semheads(@akat_hash, ftypid)[0]
723     # metoda semheads klasy FrazaAnot

725     if (synh or ftyp.moze_bez_glowy?(:synh)) and
726         (semh or ftyp.moze_bez_glowy?(:semh))
727         # jeśli nie ma którejs głowy, to fraza prawdop.
728         # jest źle oznaczona, a wtedy nic z nią nie robimy,
729         # tylko wracamy do wyboru typu.
730         fraa.vp_typ = 1 if vp and not vpneg
731         fraa.vp_typ = 0 if vp and vpneg
732         fraa.token_id_synhead = synh[1].token_id if synh
733         fraa.token_id_semhead = semh[1].token_id if semh
734         fraa.fraza_typ_id = ftypid
735         FrazaAnot.transaction do
736             fraa.save!
737             obniz_status( akapit_transzy, :syntactic )
738         end
739         # @pokaz_wybor_typu = false sam znika, ciekawe dlaczego
740     else
741         @pokaz_wybor_typu = true
742     end
743     # redirect_to :action => 'anotuj' for debug
744 else
745     logger.error( "ftypid is nil!!!!" )
746 end # of if ftypid
747 refresh_fraza( fraa.token_id_pocz, 0)
748 end # of fraza_wybor_typu

751 def fraza_wybor_anulowany
752     fraa= FrazaAnot.find(params[:id])
753     refresh_fraza( fraa.token_id_pocz, 0)
754 end

757 def fraza_glowy_ukryj
758     fraa= FrazaAnot.find(params[:id])
759     @pokaz_wybor_semh = true
760     @pokaz_wybor_synh = true
761     refresh_fraza( fraa.token_id_pocz, 0)
762 end

764 def fraza_synh_ukryj
765     fraa= FrazaAnot.find(params[:id])
766     @pokaz_wybor_synh = true
767     @pokaz_dwie_glowy = true
768     refresh_fraza( fraa.token_id_pocz, 0)
769 end

771 def fraza_semh_ukryj
772     fraa= FrazaAnot.find(params[:id])
773     @pokaz_wybor_semh = true
774     @pokaz_dwie_glowy = true
775     refresh_fraza( fraa.token_id_pocz, 0)

```

```
776     end

779     def fraza_dwie_glowy
780       fraa = FrazaAnot.find(params[:id])
781       @pokaz_dwie_glowy = true
782       refresh_fraza( fraa.token_id_pocz, 0)
783     end

785     def fraza_wybor_glowy
786       fraa = FrazaAnot.find(params[:id])
787       glowa_id = params[:frazaglowa][:glowa_id].to_i
788       glowa_id = nil if glowa_id == -1
789       ktora_g = params[:ktora_g]
790       if ktora_g == "obie" or ktora_g == "semh" : rob_semh = true
791       else rob_semh = false; end
792       if ktora_g == "obie" or ktora_g == "synh" : rob_synth = true
793       else rob_synth = false; end
794       fraa.token_id_synhead = glowa_id if rob_synth
795       fraa.token_id_semhead = glowa_id if rob_semh
796       FrazaAnot.transaction do
797         fraa.save!
798         obniz_status( akapit_transzy, :syntactic )
799       end
800       # @pokaz_wybor_synth znika samo, ciekawe czemu
801       # @pokaz_wybor_semh znika samo, ciekawe czemu
802       refresh_fraza( fraa.token_id_pocz, 0)
803     end # of fraza_wybor_glowy

806     def fraza_sie_typ_ukryj
807       fraa= FrazaAnot.find(params[:id])
808       @pokaz_wybor_sie_typu = true
809       refresh_fraza( fraa.token_id_pocz, 0)
810     end

812     def fraza_sie_typ_wybor
813       fraa= FrazaAnot.find(params[:id])
814       sie_typ = params[:frazasietyp][:sie_typ]
815       if sie_typ
816         fraa.sie_typ = sie_typ.to_i
817         FrazaAnot.transaction do
818           fraa.save!
819           obniz_status( akapit_transzy, :syntactic )
820         end
821       end
822       refresh_fraza( fraa.token_id_pocz, 0)
823     end

826     def ogladactwo
827       if r_wszewid?
828         @ile_rozbieznych = AkapitTranszy.ile_zaanotowanych( nil,
829           :rozbiezny )
830         @akapity_podsadne = Akapit.zaanotowane(
```

```

830             :podsądny ## ,
832             ## :where =>
833             "s.word_senses_uzid is not null"
834             ## :having => "count( distinct
835             s.akapit_transzy_id )=1"
836             )
837     @ile_podsadnych = AkapitTranszy.ile_zaanotowanych( nil,
838     :podsądny ) + ", razem " +
839     AkapitTranszy.ile_zaanotowanych_total( nil, :podsądny )
840     @transze = Transza.find(:all,
841     :conditions => "transza.uzytownik_id is
842     not null",
843     :include => :uzytownik,
844     :order => "uzytownik.uzytownik_id,
845     transza.transza_id" )
846     # akapity skomentowane pobieramy w partialu, bo zależy, czy mamy
847     session[:skomentowane_pokaż] czy nie
848     end
849
850     ## @akapity_zweryfikowane = Akapit.zaanotowane( :zweryfikowany
851     )
852     # nie używane
853
854     end # of ogladactwo
855
856     def quasi_transza_brutał
857     ## ( session[ :pokaz_qtra ] ||= Hash.new ) [ params[:klucz] ]
858     = true
859     @klucz = params[:klucz]
860     @qt_klucz = @klucz.split("-")
861     end # of quasi_transza_brutał
862
863     def pokaz_quasi_transze
864     ( session[ :pokaz_qtra ] ||= Hash.new ) [ params[:klucz] ] = true
865     qt_klucz = params[:klucz].split("-")
866
867     render :partial => 'quasi_transza',
868     :object => [ Akapit.quasi_transza( qt_klucz ),
869     params[:klucz] ]
870     end # of pokaz_quasi_transze
871
872     def ukryj_quasi_transze
873     if session[ :pokaz_qtra ] and session[ :pokaz_qtra ][
874     params[:klucz]]
875     session[ :pokaz_qtra ].delete( params[:klucz] )
876     end
877     render :partial => 'quasi_transza',
878     :object => [ [], params[:klucz] ]
879     end # of ukryj_quasi_transze
880
881     def ukryj_wszystkie_quasi_transze
882     session[ :pokaz_qtra ] = nil
883     redirect_to :action => :zweryf

```

```
885     end # of ukryj_wszystkie_quasi_transze
889     def prosba_o_odebranie_transzy
890       render :partial => 'odbierz_transze', :object =>
        Uzytkownik.find(params[:id])
891     end
893     def prosba_anulowana
894       render :partial => 'prosba_link', :object => params[:id]
895     end
897     def prosbe_o_odebranie_zapisz
898       # wygląda na to, że możliwość proszenia o odebranie transzy w tej wersji
899       # Anotatorni została zakryta przed oczyma śmiertelnych.
900       poo = params[:prosba_o_odebranie]
901       if poo
902         @anotator = Uzytkownik.find(params[:id])
903         tai = poo[:transza_id].to_i
904         utoc = @anotator.transze_otwarte.collect{ |tra| tra.transza_id }
905         przy = poo[:przyczyna].to_s
906         if tai and utoc.include?(tai) and przy and /[a-z]/i =~ przy
907           ProsbyAnotatorek.create(
908             :dotyczy_id => tai,
909             :dotyczy_type => "Transza",
910             :uzytkownik_id => @anotator.uzytkownik_id,
911             :rozpatrzone => false,
912             :spelniona => false,
913             :opis => przy )
914           # odebrana i tak jest 'f' by default.
915           end# of if tai ... and przy ...
916         end # of if poo
917         session[:odebrania_transz_pokaz] = true
918         render :update do |page|
919           page.replace_html( 'odbierz_transze', :partial => 'prosba_link',
920             :object => @anotator.uzytkownik_id )
921           page.replace_html( 'lista_odebranych', :partial =>
922             'lista_odebranych',
923             :object => @anotator.odebrania_transz )
924         end # of render update do page
925       end
927       def odebrania_transz_toggle(val)
928         @anotator = Uzytkownik.find(params[:id])
929         session[:odebrania_transz_pokaz] = val
930         render :partial => 'lista_odebranych', :object =>
931           @anotator.odebrania_transz
932       end
934       def odebrania_transz_ukryj
935         odebrania_transz_toggle(nil)
936       end
937     end
939     def odebrania_transz_pokaz
```

```

941     odebrania_transz_toggle(true)
942 end

945 def token_tworz_tag
946     session[:zmieniaj_dyzamb] ||= Hash.new
947     session[:zmieniaj_dyzamb][params[:id].to_i] = :tworz
948     token = Token.find(params[:id])
949     td0 = token.disamb( akat_hash )[0]
950     if td0 :     @lemat = td0.leksem.lemat
951     else # tokeny mogą nie być zdzyzambiguowane
952         @lemat = if int0 = token.interpretacja[0]
953                 int0.leksem.lemat
954                 else token.orth
955                 end
956     end
957     render :partial => 'token_tworz_tag', :object => [ token,
958               @akapit_transzy ]
959 end# of token_tworz_tag.

961 def autocomplete_tag
962     currval =
963     params["token_tag#{params[:id]}"][:tag].to_s.strip.downcase
964     # downcase added 2010/6/16 to fix #356
965     ##      re = Regexp.new("^#{currval}" , "i" )
966     ##      @tags= Tagset.znajdz_pos( :all ).collect{ |t|
967     ##          t.lewe if t.lewe.match re
968     ##      }.compact
969     ##      @tags = [ 'brekekekk', 'keke brekk' ]
970     @tags = Tagset.matches( currval )
971     ##      logger.info "#### " + @tags.inspect
972     render :layout => false
973 end

975 def interpretacja_z_tagu
976     set_uzid
977     @tokid = params[:id].to_i
978     tagparam = "token_tag#{@tokid}".intern
979     lematparam="token_lemat#{@tokid}".intern
980     logger.info "#### przed tworzeniem tagu: params=#{params.inspect}"
981     tag = ( params[tagparam] ||
982             (logger.info "#### params[#{tagparam.inspect}] is nil!"
983               {}) )[:tag].to_s.
984     downcase # added 2010/6/16 to fix #356
985     lemat = params[lematparam][:lemat]
986     ct = Tagset.check_tag( tag, lemat )
987     if ct[0]
988         iid = Interpretacja.fc_by_fulltag( @tokid, lemat, tag ).id
989         conds = {
990             :akapit_transzy_id => akapit_transzy.id,
991             :token_id => @tokid
992         }
993     end

```



```

999      ia = InterpretacjaAnot.find( :first, :conditions => conds )
1000      ia ||= InterpretacjaAnot.new( conds )
1001      ia.uzytownik_id = @uzid
1002      ia.interpretacja_id = iid
1003      ia.save!
1004      Interpretacja.odsmiec( :token_id => @tokid )
1005      Token.find( @tokid ).zambiguuj

1007      if ( szd = session[ :zmieniam_dyzamb ] ) and szd[ @tokid ]
1008        szd[@tokid] = nil
1009      end
1010      odswiez_token( @tokid )
1011    else flash.now[:tworz_tag] = ct[1]
1012      session[ :tworz_tag ] ||= Hash.new
1013      session[ :tworz_tag ][ @tokid ] = { :lemat => lemat, :tag =>
1014        tag }
1014      odswiez_token( @tokid, :tworz_tag )
1015    end
1016  end# of interpretacja_z_tagu.

1019  def nowytag_skopiuj_orth
1020    session[ :tworz_tag ] ||= Hash.new
1021    @tokid = params[ :id ].to_i
1022    lemat = Token.find( @tokid ).orth
1023    session[ :tworz_tag ][ @tokid ] ||= Hash.new
1024    session[ :tworz_tag ][ @tokid ].update( @tokid => { :lemat =>
1025      lemat } )

1026    render :update do |page|
1027      page.replace_html "token_lemat#{@tokid}", :partial =>
1028        'tworz_tag_lemat',
1029      :object => [ @tokid, lemat ]
1029    end

1031  end # of nowytag_skopiuj_orth

1034  def poziomuj
1035    set_uzid
1036    parampa = params[ :poziomy_anotacji ]
1037    if parampa
1038      ## logger.info( parampa.class )
1040      set_poziomy
1041      parampa.each_pair{ |key, val|
1042        poziom = key.intern
1043        ## logger.info( "=== poziom #{poziom}:
1044          #{val.to_bool}" )
1045        if PoziomyAnotacji.działające.include?( poziom )
1046          @poziomy_wybrane.set_poziom( poziom, val.to_bool ) ##
1048          unless poziom == :morphosyntactic
1049            # AP chce móc wyłączyć morfosyntaks.
1049          end
1050        }

```

```

1051         @poziomy_wybrane.save!
1053         redirect_to :action => :anotuj, :akapit => akapit_transzy, :anchor
            => :primafalsa
1054     end
1055 end # of poziomuj.

1058 def zarządca_jak_anotator
1059     zja = params[:jak_anotator][:jak_anotator].to_bool
1060     session[ :jak_anotator ] = zja
1061     set_uzid
1062     @użytkownik.jak_anotator = zja
1063     @użytkownik.save!

1065     redirect_to :action => :anotuj, :akapit => akapit_transzy, :anchor
        => :primafalsa
1066 end

1069 # akcje związane z ujednoznacznieniem segmentacji

1071 def segmentacja_zmieniaj
1072     set_poziomy
1073     @tokid = params[:id].to_i
1074     ( session[ :zmieniaj_segmentacje ] ||= {} )[ @tokid ] = true
1075     odswiez_token(@tokid, :segmentation)
1076 end # of segmentacja_zmieniaj

1079 def segmentacja_ujednoznacznij
1080     set_poziomy
1081     segdec=params[:sg_variant_anot]
1082     ##         logger.info( "### segdec: #{segdec}, class:
        #{segdec.class}" )
1084     session[:zmieniaj_segmentacje] ||= Hash.new
1085     if segdec
1086         segdec=segdec[ :wybierz ].to_bool
1087         # teraz segdec jest true gdy mamy ten wybrać, bądź false gdy mamy
            ten odrzucić
1088         tokid = params[:id].to_i
1089         token = Token.find( tokid )
1090         sg_choice = token.sg_choice
1091         sgvid = token.sg_variant_id
1092         sguj_tokids = []
1093         SgVariantAnot.transaction do
1094             sg_choice.sg_variant.each{ |sgv|
1095                 sgv_cond = {
1096                     :akapit_transzy_id => akapit_transzy.id,
1097                     :sg_choice_id => sg_choice.id,
1098                     :sg_variant_id => sgv.id
1099                 }
1100                 sgvan = SgVariantAnot.find(
1101                     :first,
1102                     :conditions => sgv_cond)
1103                 sgvan ||= SgVariantAnot.new( sgv_cond )

```

```

1104      # oznaczamy wariant jako wybrany wtw gdy
1105      sgvan.chosen = if sgvid == sgvid
1106                     segdec
1107                     else ( not segdec )
1108                     end
1109      sgvan.save!
1110      sg.token.each{ |tok|
1111        if sgvan.chosen? and not tok.chosen?
1112          tok.chosen =true
1113          tok.save!
1114        end
1115        sguj_tokids « tok.id }
1116      }# of each segm. variant
1117      obniz_status( akapit_transzy, :segmentation )
1118    end# of transaction
1119  else# not intid
1120    sguj_tokids = [ params[:id].to_i ]
1121  end
1122  #      redirect_to :action => 'anotuj' for debug
1123  sguj_tokids.each{ |tokid|
1124    raiseifzero( tokid, "segmentacja_ujednoznaczniej" )
1125    session[ :zmieniaj_segmentacje ][ tokid ] = nil }
1126
1127    odswiez_token( sguj_tokids, :segmentation )
1128
1129  end # of segmentacja_ujednoznaczniej.
1130
1131  def segmentacja_anulowana
1132    @tokid=params[ :id ].to_i
1133    set_poziomy
1134    szs = session[ :zmieniaj_segmentacje ]
1135    szs[@tokid] = nil if szs
1136    raiseifzero( @tokid, "segmentacja_anulowana" )
1137    odswiez_token( @tokid, :segmentation )
1138  end # of segmentacja_anulowana
1139
1140  # granice zdań
1141
1142  def konczymy_zdanie
1143    @tokid=params[ :id ].to_i
1144    set_poziomy
1145    akat=akapit_transzy
1146    kz = KoniecZdaniaAnot.znajdz( akat, @tokid )
1147
1148    if kz
1149      kz.destroy
1150    else
1151      KoniecZdaniaAnot.create(
1152        :akapit_id => akat.akapit.id,
1153        :akapit_transzy_id => akat.id,
1154        :uzytkownik_id => akat.get_uzid( session[
1155          :uzid ] ),
1156        :token_id => @tokid

```

```

1158                                     )
1159     end
1160     odswiez_token( @tokid, :sentences )
1161 end # of konczymy_zdanie.

1164 def nowa_prośba_segmentacyjna
1165   if params[ :akapit ]
1166     ak = Akapit.find( params[ :akapit ] )
1167   elsif r_annotator?
1168     raise "Prośba o nową segmentację bez akapitu!
1169           (anotacja_controller.rb l. 1005)"
1170   end
1171   if seg = params[ :nowa_segmentacja ]
1172     seg_is = seg[ :is ].split
1173     seg_should_be = seg[ :should_be ].split
1174     nps = true # domyślnie ma być bez spacji
1175     if seg_should_be.size > 1
1176       # jeśli nie zaznaczono opcji dot. spacjowania, przyjmujemy, że ma
1177       # być bez spacji
1178       nps = ( seg[ :nps ] == nil ) || seg[ :nps ].to_bool
1179     end
1180     seg_opis = seg[ :opis ]
1181     seg_akids = seg[ :akids ]
1182   end

1183   ##      logger.info "=== nps: #{nps.inspect}"
1184   ##      logger.info "=== akapit #{params[:akapit].inspect}"
1185   ##      logger.info "=== params[ :nowa_segmentacja ] ==
1186   ##                  #{params[:nowa_segmentacja].inspect}"
1187   ##      logger.info "=== seg_is.inspect == #{seg_is.inspect}"
1188   # chcemy ustalić akapit, w którym występują takie token(y)
1189   if ak and ak.sa_orthy?( seg_is, :strict )
1190     ids = [ ak.id ]
1191   elsif seg_akids and (akids = ( seg_akids.split.collect{ |akid|
1192     akid.to_i } - [0] ))[0]
1193     ids = []
1194     akids.each { |id214|
1195       if a214 = Akapit.find( id214 ) and a214.sa_orthy?( seg_is,
1196         :strict )
1197         ids << id214
1198       end
1199     }
1200   else
1201     ids = Akapit.znajdź_po_orthach( seg_is, :strict ).collect{ |x|
1202       x.id }
1203   end

1204   if ak
1205     unless ids.include?( ak.id )
1206       notice = "Podane tokeny nie występują w akapicie #{ak.id}. Nic
1207       nie robię."

```

```

1208     else
1209         notice = nil
1210     end
1211     else # not ak czyli nie podano akapitu, w którym miałyby to być
1212         if ids[0]
1213             notice = nil
1214         else
1215             notice = "W akapicie/ach #{ids.join(', ')} nie ma takiego
                układu tokenów. Nic nie robię."
1216         end
1217     end
1219     unless notice
1220         # sprawdzimy, czy rzeczywiście chodzi o zmianę segmentacji
1221         if seg_is.join != seg_should_be.join
1222             notice = "Podany stan obecny (»#{seg_is.join(' ')}«) nie zgadza
                się ze stanem żądanym (»#{seg_should_be.join(' ')}«). Nic nie
                robię."
1223         elsif seg_is == seg_should_be
1224             notice = "Proponujesz to samo, co już jest (w obu polach
                »#{seg_is.join(' ')}«). Nic nie robię."
1225             # nie odrzucamy przypadku »ala ma kota« -> »ala makota« tzn. gdy
                któryś token pozostaje niezmieniony.
1226         elsif NowaSegmentacja.find( :first,
                :conditions => (
1227                                 ns_hash194 = {
1228                                     :seg_is => seg_is.to_yaml,
1229                                     :seg_should_be =>
1230                                         seg_should_be.to_yaml,
1231                                     :ids => ids.to_yaml,
1232                                     :nps => nps } ) )
1233             notice = "Już zgłoszono taką prośbę o zmianę segmentacji
                »#{seg_is.join(' ')}«. Nic nie robię."
1234         elsif NowaSegmentacja.new( ns_hash194 ).instanze( :seg_should_be
                )[ 0 ]
1235             notice = "Takie »ma być« już <b>*jest*</b>. Nic nie robię."
1236         end
1237     end
1239     if notice
1240         flash.now[ :nowa_segmentacja ] = notice
1241     else
1242         ns_hasz = { :ids => ids,
1243                     :seg_is => seg_is,
1244                     :seg_should_be => seg_should_be,
1245                     :nps => nps
1246                 }
1247         if pakat = params[ :akat ]
1248             akat = AkapitTranszy.find( pakat )
1249             ns_hasz.update(
1250                 :akapit_transzy_id => akat.id,

```

```

1251             :transza_id => akat.transza_id )
1252     end
1253
1254     p = ProsbyAnotatorek.create!(
1255         :uzytkownik_id => ( session[
1256             :drag_queen_id] || session[:uzid] ),
1257         :dotyczy => NowaSegmentacja.new( ns_hasz
1258             ),
1259         :opis => seg_opis
1260     )
1261     flash.now[ :nowa_segmentacja ] = "Zapisałam prośbę o
1262     #{p.treść_acc}."
1263 end
1264
1265     prośby_toggle( :nowa_segmentacja_pokaz, nil )
1266     refresh_prośby
1267 end # of nowa_prośba_segmentacyjna
1268
1269 def nowa_segmentacja_ukryj
1270     akapit_transzy if r_anotator?
1271     prośby_toggle( :nowa_segmentacja_pokaz, nil )
1272     render_prośby
1273 end
1274
1275 def nowa_segmentacja_pokaz
1276     akapit_transzy if r_anotator?
1277     prośby_toggle( :nowa_segmentacja_pokaz, true )
1278     render_prośby
1279 end
1280
1281 def spełnij_prośbę
1282     prośba = ProsbyAnotatorek.find(params[:id])
1283     unless prośba.dotyczy_type == params[:dotyczy_type] and
1284         prośba.dotyczy_id == params[:dotyczy_id].to_i
1285         flash[:prosby_anotatorek] = "Prośba przekazana w parametrach
1286         niezgodna z prośbą zapisaną w bazie!"
1287     else # prośba zgodna z tym co w bazie
1288         if p = prośba.commit( session[ :rola_id ] )
1289             flash[ :prosby_anotatorek ] = p
1290         end
1291         refresh_prośby( prośba.dotyczy.is_a?( Transza ))
1292     end # of prośba check
1293 end # of rozpatrz_prośbę
1294
1295 def dodaj_segmentacje # 2009/6/24 prawdopodobnie niepotrzebna: nie
1296     jest używana.
1297     @akapit = Akapit.find( params[:akapit_id] )
1298     @prośba = ProsbyAnotatorek.find( params[:prosba_id])
1299 end# of dodaj_segmentacje
1300
1301 def odrzuć_prośbę
1302     flash.now[ :prosby_anotatorek ] =
1303     ProsbyAnotatorek.find( params[:id] ).odrzuć( session[ :rola_id ]
1304     )

```

```
1304     refresh_prośby
1305   end # of odrzuć_prośbę

1308   def sense_inventory
1309   end

1311   def zweryf # używana przez Beatę
1312     @akapity_zweryfikowane = Akapit.zaanotowane( :zweryfikowany )
1313   end # of zweryf

1317   def wsze_transze_otwarte_toggle
1318     if params[:direction] == 'show'
1319       session[:pokaż_wsze_transze_otwarte] = true
1320     else
1321       session[:pokaż_wsze_transze_otwarte] = false
1322     end

1324     render :update do |page|
1325       page.replace 'wsze_transze_otwarte',
1326         :partial => 'wsze_transze_otwarte'
1327     end
1328   end # of wsze_transze_otwarte_toggle

1331   def wsze_transze_BW_toggle
1332     if params[:direction] == 'show'
1333       session[:pokaż_wsze_transze_BW] = true
1334     else
1335       session[:pokaż_wsze_transze_BW] = false
1336     end

1338     render :update do |page|
1339       page.replace 'wsze_transze_BW',
1340         :partial => 'wsze_transze_BW'
1341     end
1342   end # of wsze_transze_BW_toggle

1345   private

1348   @akapit_transzy = nil
1349   @akapit_bliźniak = nil

1352   def akapit_transzy
1353     set_uzid

1355     if ( pakat = params[ :akapit ] ) and ( pakat.to_i != session[ :akapit ] )
1356       zniluj_akapitowe
1357       session[ :akapit ] = pakat.to_i
1358       @akapit_transzy = nil
1359     end

1361     unless @akapit_transzy

1363       if session[ :akapit ]
1364         @akapit_transzy = AkapitTranszy.find( session[ :akapit ] )
1365       else
```

```

1366      ##      logger.info "#### no session[ :akat ] anotator:
1367      #{session[:anotator]}"
1368      @akapit_transzy = AkapitTranszy.znajdz( session[:akapit],
1369      @anotid )
1369      end
1370      end# of unless @akapit_transzy
1371
1372      @akapit_transzy.reload # bo status się nie odświeżał
1373
1374      session[ :akat ] = @akapit_transzy.id
1375
1376      unless @akat_bliźniak
1377        @akat_bliźniak = @akapit_transzy.bliźniaczy
1378      end
1379
1380      if ( not @transza ) or @transza.id != @akapit_transzy.transza_id
1381        @transza = @akapit_transzy.transza
1382        if r_wszewid? and session[ :anotator ] !=
1383          @akapit_transzy.uzytownik_id and
1384            session[ :anotator ] != @transza.uzytownik_id
1385          @anotid = ( session[ :anotator ] = @transza.uzytownik_id )
1386        end
1387      end
1388
1389      ## # logger.info "##### akapit transzy ustalony: @akapit_transzy.id"
1390      @akapit_transzy
1391      end # of akapit_transzy
1392
1393      def akat_hash
1394        unless @akat_hash
1395          @akat_hash = akapit_transzy.akat_hash
1396        end
1397        @akat_hash
1398      end
1399
1400      def akat_walencyjny?
1401        Walencja.x and ( @akapit_transzy.ramka_id or
1402          (@akapit_transzy.ma_status?( :>=, :doramki ) and
1403            (not @akapit_transzy.niefrazowny?) and
1404            not @akapit_transzy.niesensowny?))
1405      end
1406
1407      def inicjuj_akat
1408        akapit_transzy
1409        akat_hash
1410      end
1411
1412      def odswiez_token( * params )
1413        tokids = params.delete_at(0)
1414        ## logger.info "##### tokids w odśwież_token:
1415        #{[tokids].flatten.join(', ')}"
1416        tokens = Token.find( :all,
1417          :conditions => { :token_id => [tokids].flatten,
1418            :chosen => true } # drugi warunek dodany
1419          2009/8/23 w ramach tropienia #246.

```



```

1419          ) # musi umieć obsłużyć tablicę idów, ze względu na
              segmentacja_ujednoznaczni_j.

1420      set_uzid
1421      akat_hash
1422      render :update do |page|
1423        if not ( params & [ :tylko_sens, :segmentation, :sentences ] )[ 0
1424          ]
1425          if params.include?( :tworz_tag )
1426            tokpartial = 'token_tworz_tag'
1427          else
1428            tokpartial = 'token_disamb'
1429            session[:tworz_tag] = nil if session[:tworz_tag]
1430          end
1431          tokens.each{ |tok|
1432            page.replace_html
1433              "disa-token_#{@akapit_transzy.id}_#{tok.id}",
1434              :partial => tokpartial, :object => [tok, @akapit_transzy,
1435                true] }

1436        elsif (params & [ :segmentation, :sentences ])[ 0 ]
1437          tokens.each{ |tok|
1438            page.replace
1439              "outer_seg-token_#{@akapit_transzy.id}_#{tok.id}",
1440              :partial => 'token_td', :object => [tok, @akapit_transzy,
1441                true] }
1442        else
1443          tokens.each{ |tok|
1444            page.replace_html
1445              "sens-anot_#{@akapit_transzy.id}_#{tok.id}",
1446              :partial => 'token_sens', :object => [tok, @akapit_transzy,
1447                true] }
1448        end

1449        if params.include?( :sentences )
1450          page.replace_html(
1451            'sentensnik',
1452            :partial => 'sentensnik', :object =>
1453              @akapit_transzy
1454            # z jakiegoś powodu nie działało, kiedy wpisywałem tu
1455            akapit_transzy
1456          )
1457        end

1458        page.replace_html(
1459          'status_info',
1460          :partial => 'status_info'
1461          # z jakiegoś powodu nie działało, kiedy wpisywałem tu
1462          akapit_transzy
1463        )
1464        page.replace_html(
1465          'zatwod_buttons',

```

```

1459             :partial => 'anotuj_zatwod_nk'
1460             # z jakiegoś powodu nie działało, kiedy wpisywałem tu
             akapit_transzy
1461         )
1462     end
1463 end# of odswiez_token.

1466 def refresh_fraza(tokpocz, tokkon)
1467     # musi być tokpocz<=tokkon, inaczej zrenderuje się tylko dla tokpocz
1468     tokkon = tokpocz if tokpocz > tokkon
1469     refresh_walencje if Walencja.x
1470     akat_hash

1472     render :update do |page|
1473         tokpocz.upto(tokkon) do |tokid|
1474             page.replace( # replace_html zastępuje wewnątrz eltu, ta — cały elt
1475                 'token_fra_' + tokid.to_s,
1476                 :partial => 'token_fra',
1477                 :object => [Token.find(tokid), @akat_hash, :lewa])
1478             page.replace( # replace_html zastępuje wewnątrz eltu, ta — cały elt
1479                 'framka_' + tokid.to_s,
1480                 :partial => 'token_framka',
1481                 :object => [Token.find(tokid), @akat_hash])
1482         end
1483         page.replace_html(
1484             'status_info',
1485             :partial => 'status_info',
1486             :object => @akapit_transzy
1487             # z jakiegoś powodu nie działało, kiedy wpisywałem tu
             akapit_transzy
1488         )
1489         page.replace_html( 'walencja', :partial => 'walencja')
1490     end
1491 end# of refresh_fraza

1494 def zniluj_akapitowe( * keys )
1495     ## logger.info "### zniluj akapitowe"
1496     klucze = keys + [ :akapit, :akat, :status, :zdziw_sie_akapitowi ]
1497     # 2009/04/18 usunąłem z powyższej listy klucz :weryfikacja.
1498     klucze += [ :zmieniaj_dyzamb,
1499                 :tworz_tag,
1500                 :zmieniaj_segmentacje] if DlaEli.nk
1501     klucze.each{|klucz| session[ klucz ] = nil }
1502     # session[:anotator] = nil
1503 end# of zniluj_akapitowe

1507 def podnies_status( akat, poziom, weryfikacja=nil,
czy_bliźniaczy=nil )
1508     # ostatni argument jest podany jako nie-nil tylko przy przyklepywaniu
    akapitu.

```

```

1509      ##      logger.info "####
##{annotation_controller.capitalize}Controller.podnies_status:
akat #{akat.id} \n" +
1511      "poziom: #{poziom.inspect}, weryfikacja:
#{weryfikacja.inspect}, \n " +
1512      " czy_bliźniaczy: #{czy_bliźniaczy.inspect}"
1513      akat_STATI = AkapitTranszy::STATI
1514      stati_pdn = AkapitTranszy::STATI_podnies# hasz kodujący graf
podnoszenia poziomu
1515      akats = akat.statusy
1516      blis = akat.bliźniaczy.statusy

1518      if r_zarzacca?
1519          akatsp = AkapitTranszy::STATI[:osadzany]
1520          czy_bliźniaczy = true
1521      else
1522          akatsp = akats.send( poziom )
1523      end
1524      ##      logger.info "#### podnies_status, weryfikacja:
>#{session[:weryfikacja].inspect}<, czy bliźniaczy:
#{czy_bliźniaczy.inspect}\n"
1526      if ( not ( weryfikacja and
1527          ( werdykcik = # (jedno !=)
weryfikacja[:werdykciki][poziom] )
1528          # taki układ spójników logicznych, żeby koniecznie obliczyć
werdykcik, bo potrzebujemy go dalej
1529          )) and stati_pdn.has_key?( akatsp )
1530          nowy_statusek = stati_pdn[ akatsp ]
1531          Statusy.transaction( akats, blis ) do
1532              akats.set_poziom( poziom, nowy_statusek )
1533              akats.save!
1534              if czy_bliźniaczy or nowy_statusek >= akat_STATI[ :do_osadzenia
1535              ]
1536                  ##      logger.info "#### ustawiam bliźniaczy"
1537                  blis.set_poziom( poziom, nowy_statusek )
1538                  blis.save!
1539              end
1540          end # of transaction
1541      elsif werdykcik
1542          # (Może być nieznilowany werdykcik dla poprzedniego poziomu, a dla
naszego może jeszcze nie być.)
1543          # w tym wypadku uwzględnimy werdykcik:
1544          Statusy.transaction( akats, blis ) do
1545              nowy_statusek = stati_pdn[ [akatsp, werdykcik] ]
1546              akats.set_poziom( poziom, nowy_statusek )
1547              akats.save!
1548              if czy_bliźniaczy or nowy_statusek >= akat_STATI[ :do_osadzenia
1549              ]
1550                  ##      logger.info "#### ustawiam bliźniaczy"
1551                  blis.set_poziom( poziom, nowy_statusek )

```

```

1553         blis.save!
1554     end
1555     end# of transaction
1556 else
1557     raise( "#{annotation_controller}_controller.podnies_status: ani
1558           weryfikacji, ani gołego statuska w haszu???" )
1559 end

1560 # ten akapit jest skopiowany z metody
1561 <akapit_transzy>.status_co_najmniej=
1562 # która zapewne jest tym sposobem niepotrzebna.
1563 if akat.statusy.send( poziom ) < akat_STATI[ :zweryfikowany ] and
1564     akat.bliźniaczy.statusy.send( poziom ) >= akat_STATI[
1565         :zweryfikowany ]
1566     blis.set_poziom( poziom, akat_STATI[ :osądzany ] )
1567     blis.save!
1568     ##         logger.info( "=== podnies_poziom: obniżyłam status
1569         bliżn. na poz. :#{poziom} do :osądzany." )
1570 end
1571 end# of podnies_status.

1572 def obniz_status( akat, poziom )
1573     akats = akat.statusy
1574     akatsp = akats.send( poziom )
1575     stati = AkapitTranszy::STATI_obniz
1576     ##         logger.info "=== tu działa #{session[ :login ].inspect}"
1577     ##         logger.info "=== obniżam status akat. #{akat.id}
1578         #{poziom.inspect} z #{akatsp} na #{stati} (akat.status był
1579         #{akat.status})"
1580     Statusy.transaction do
1581         akats.set_poziom( poziom, stati[akatsp] ) # jeśli akapit było na
1582         tym poziomie zakończony, a teraz obniżamy status, to musimy zakazać
1583         anotacji na wyższych poziomach, i ta metoda to robi.
1584         akats.save!
1585         akat_STATI = AkapitTranszy::STATI
1586         if akats.send( poziom ) == akat_STATI[ :osądzany ]
1587             blis = akat.bliźniaczy.statusy
1588             blisp = blis.send( poziom )
1589             if blisp > akat_STATI[ :osądzany ]
1590                 blis.set_poziom( poziom, akat_STATI[ :osądzany ] )
1591                 blis.save!
1592             end # of if bli.status
1593         end # of akat.satus == osądzany
1594     end # of transaction
1595     session[ :status ] = akat.status
1596     ##         logger.info "=== akat. #{akat.id} .status ==
1597         #{akat.status}"
1598 end

1599 def przypisz_oba_uzidy( poziom )
1600     akats = akapit_transzy.statusy.reload

```

```

1601      uzid = @akapit_transzy.get_uzid( session[ :uzid ] )
1603
1603      blis = @akat_bliźniak.statusy.reload
1604      bluzid = @akat_bliźniak.get_uzid( session[ :uzid ] )
1606
1606      Statusy.transaction( akats, blis ) do
1607        akats.set_uzid( poziom, uzid )
1608        blis.set_uzid( poziom, bluzid )
1609        akats.save!
1610        blis.save!
1611      end # of transaction
1612    end# of przypisz_oba_uzidy
1615
1615    def zweryfikuj_akapit( akat, destiny, poziomy, notice=nil )
1616      ##      logger.info "=== zweryfikuj_akapit 1164: poziomy:
1616      ##      logger.info "                               statuski:
1618      ##      logger.info "                               #{akat.statuski.inspect}"
1621
1621      pd = akat.poziomy_dostępne(
1622        session[:rola_id],
1623        :jak_anotator => session[ :jak_anotator ] )
1624      ##      logger.info "=== statuski: >#{akat.statuski.inspect}<,
1624      ##      logger.info "                               rola_id: >#{session[:rola_id]}<, pd: >#{pd.inspect}<"
1626      pd &= poziomy
1628
1628      ##      logger.info "=== po weryfikacji 1476: akat:
1628      ##      logger.info "                               #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
1628      ##      logger.info "                               #{akat.bliźniaczy.statusy.reload.statuski.inspect}"
1631
1631      weryf = akat.weryfikacja(
1632        :dokonca => true,
1633        :protokoluj => true, # protokoluj rozbieżności
1634        :poziomy => pd,
1635        :rola_id => session[:rola_id],
1636        :właśnie_zatwierdzamy => true,
1637        :session_uzid => session[ :uzid ]
1638      )
1640
1640      session[:weryfikacja] = weryf
1642
1642      ##      logger.info "=== po weryfikacji 1489: akat:
1642      ##      logger.info "                               #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
1642      ##      logger.info "                               #{akat.bliźniaczy.statusy.reload.statuski.inspect}"
1645
1645      ver = {}
1647
1647      # procedura gdy wszystko gra i bucy:
1648      przyklep = Proc.new { |poz|
1649        na_poz = "Na poz. #{PoziomyAnotacji.gen( poz )}"
1650        ver[ poz ] = "#{na_poz} przeszedł on weryfikację, gratuluję." if
1651        destiny == :zatwierdź
1651        ver[ poz ] = "#{na_poz} Twój partner także go odrzucił.
1651        Gratuluję." if destiny == :odrzuć

```

```

1653      ##      logger.info "=== po weryfikacji 1494: akat:
      #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
      #{akat.bliźniaczy.statusy.reload.statuski.inspect}"

1656      podnies_status( akat, poz, weryf, :bliźniacze_też )
1657      ##      logger.info "=== po podnies_status 1502: akat:
      #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
      #{akat.bliźniaczy.statusy.reload.statuski.inspect}"

1659      przypisz_oba_uzidy( poz )
1660      ##      akat.set_status(poz, :zweryfikowany) if r_anotator?
1662      ##      akat.set_status(poz, :osądzony) if r_zarządca?
1664      ##      logger.info "=== zweryfikuj_akapit 1.1506 (przyklep),
      akat. #{akat.id}: #{akat.statuski.inspect} bliźniaczy:
      #{akat.bliźniaczy.statuski.inspect}"

1666      akat.send( "commit_#{poz}" )
1667      # na poziomach seg i sent, których zatwierdzenie zmienia treść (tekst)
      akapitu przeznaczoną do wyświetlania, metoda commit... aktualizuje ją.

1668      ##      logger.info "=== po weryfikacji 1500: akat:
      #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
      #{akat.bliźniaczy.statusy.reload.statuski.inspect}"

1670      Protokol.superancje( poz, akat, session[:uzid] ) if r_zarządca?
1671      ##      logger.info "=== po weryfikacji 1511: akat:
      #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
      #{akat.bliźniaczy.statusy.reload.statuski.inspect}"

1673    } # of przyklep

1675    # podprocedura w wypadku niezrobienia przez partnera
1676    no_verdict = Proc.new { |poz|
1677      na_poz = "Na poz. #{PoziomyAnotacji.gen( poz )}"
1678      obrobił = "zaanotował"
1679      # tu ma być zależnie od statuska
1680      podnies_status( akat, poz, weryf )
1681      ##      logger.info "=== podnies_status in zweryfikuj_akapit 1197"
1683      obrobił = "poprawił" if akat.ma_status?( poz, ==, :po_poprawce )
1684      ##      logger.info "zweryfikuj_akapit 1178, akat.
      #{akat.id}: #{akat.statuski.inspect}"

1686      ver[ poz ] = "#{na_poz} Twój partner jeszcze go nie #{obrobił}
      &#0151; weryfikacja odłożona."

1687    } # of no_verdict

1689    nie_poprawki = true
1690    bli_po_poprawce = true

1692    # podprocedura gdy źle
1693    condemn = Proc.new { |poz|
1694      na_poz = "na poz. #{PoziomyAnotacji.gen( poz )}"
1695      nie_poprawki = true
1696      nie_poprawki && akat.ma_status?( poz, <, :do_poprawki )
1697      ##      if (not session[:weryfikacja]) or
      session[:weryfikacja][:werdykt]==0

1699      if nie_poprawki
1700        akat.set_status( poz, :do_poprawki )

```

```

1701      ##      logger.info "zweryfikuj_akapit 1188, akat.
      #{akat.id}: #{akat.statuski.inspect}"
1703      ver[ poz ] = "Akapit #{akat.akapit_id} poddany weryfikacji
      #{na_poz} nie przeszedł jej. Do poprawki." if destiny ==
      :zatwierdz
1704      ver[ poz ] = "Twój partner zaanotował akapit #{akat.akapit_id}
      #{na_poz} i&nbsp;zatwierdził. Do poprawki." if destiny ==
      :odrzuc
1705      else
1706        bli_po_poprawce = true
1707        bli_po_poprawce &&= akat.bliźniaczy.ma_status?( poz, :>=,
        :po_poprawce )
1708        if bli_po_poprawce
1709          akat.set_status( poz, :do_osądzenia )
1710          ##      logger.info "zweryfikuj_akapit 1200, akat. #{akat.id}:
          #{akat.statuski.inspect}"
1712          ver[ poz ] = "Akapit #{akat.akapit_id} poddany ponownej
          weryfikacji #{na_poz} nie przeszedł jej. Przechodzi pod sąd
          Superanotatora." if destiny == :zatwierdz
1713          ver[ poz ]= "Twój partner zaanotował akapit #{akat.akapit_id}
          #{na_poz} i&nbsp;zatwierdził ponownie. Pod sąd
          Superanotatora." if destiny == :odrzuc
1714        else # nie bli_po_poprawce
1715          akat.set_status( poz, :po_poprawce,
1716                        unless Rola.zarzadca?( session[:rola_id ] )
1717                          :tylko_to
1718                        else nil
1719                      end
1720                    )
1721          ##      logger.info "zweryfikuj_akapit 1207, akat.
          #{akat.id}: #{akat.statuski.inspect}"
1723          ver[ poz ] = "Twój partner jeszcze nie poprawił anotacji
          #{na_poz} &#0151; weryfikacja odłożona."
1724        end # of if bli_po_poprawce
1725      end # of if nie_poprawki.
1726    } # of condemn

1728    case weryf[:werdykt]
1729    when 1
1730      poziomy.each{ |poz| przyklep.call( poz ) }
1731      ##      logger.info "%% po weryfikacji 1557: akat:
      #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
      #{akat.bliźniaczy.statusy.reload.statuski.inspect}"
1733    else ##      when 0
1735      poziomy.each{ |poz|
1736        case weryf[ :werdykciki ][ poz ]
1737        when 1
1738          przyklep.call( poz )
1739        when 0
1740          no_verdict.call( poz )

```

```

1741         when -1
1742             condemn.call( poz )
1743         end
1744     }
1745     ##      logger.info "=== po weryfikacji 1569: akat:
        #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
        #{akat.bliźniaczy.statusy.reload.statuski.inspect}"
1747 end# of case weryf[:werdykt]
1748 ## end

1751     ##      logger.info "=== po weryfikacji 1572: akat:
        #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
        #{akat.bliźniaczy.statusy.reload.statuski.inspect}"

1754     ver_total = ( [ notice ] + poziomy.collect { |poz| ver[ poz ]
        }).compact.join( "<br>\n")

1756     if weryf[:werdykt] == -1
1757         flash[:notice] = "Zażądałeś/eś zatwierdzenia akapitu
        #{akat.akapit.id}:\n <br><i style=\"font-weight:
        normal\">#{akat.akapit.tresc}</i><br>" +
1758         ver_total
1759         if nie_poprawki or bli_po_poprawce
1760             redirect_to :action => :anotuj, :akat => akat, :anchor =>
                :notizia
1761         else
1762             zniluj_akapitowe
1763             redirect_to :action => :pokaz_transze
1764         end
1765     else # weryf[:werdykt] != -1
1766         flash[:notice] =
1767             "#{destiny}iłeś/eś akapit #{akat.akapit.id}:\n <br><i
                style=\"font-weight: normal\">#{akat.akapit.tresc}</i><br>" +
                ver_total
1768
1769         if weryf[ :werdykt ] == 1 and akat.ma_Status?( :<, :zatwierdzony
                )
1770             # jeżeli akapit na tych poziomach się zweryfikowało, a jeszcze jest
                w nim co anotować, to renderujemy stronę jego anotacji.
1771             ##      logger.info "==== redirecting to anotuj"
1772             redirect_to :action => :anotuj, :akat => akat
1773         else
1774             zniluj_akapitowe
1775             redirect_to :action => :pokaz_transze if r_anotator?
1776             redirect_to :action => :ogladaactwo if r_zarzadca?
1777         end
1778     end
1779 end
1780     ##      logger.info "=== po weryfikacji 1597: akat:
        #{akat.statusy.reload.statuski.inspect}, bliźniaczy:
        #{akat.bliźniaczy.statusy.reload.statuski.inspect}"
1782 end # of zweryfikuj_akapit,
1784 @anotid=nil

```



```

1785     @uzid=nil
1786     @uzytkownik=nil

1789     def set_uzid
1790         @anotid = ( session[ :anotator ] || session[ :uzid ] ) unless
            @anotid
1791         @anotid = params[ :anotator ].to_i if params[ :anotator ]
1792         @uzytkownik = Uzytkownik.find( session[ :uzid ] ) unless
            @uzytkownik
1793         @uzid = ( session[:uzid] ) unless @uzid
1794         @uzid
1795     end

1798     def set_pozioomy
1799         inicjuj_akat
1800         set_uzid
1801         @poziomy_wybrane = PoziomyAnotacji.znajdz_u( @uzid )
1802         @poziomy = @poziomy_wybrane.z_pokaznymi( @akapit_transzy, session[
            :rola_id ] )
1803         session[ :poziomy ] = @poziomy.trues
1804         # to się znajdzie, bo w migracji o42 zapełniamy tablicę idami
            użytkowników.
1805         ##      logger.info "#### wybrane:
            #{@poziomy_wybrane.inspect}\n\t\t z_pokaznymi:
            #{@poziomy.inspect}"
1807     end # of set_pozioomy

1809     alias :inicjuj_akat_i_pozioomy :set_pozioomy

1812     if Walencja.x
1813         def refresh_walencje( with_render=nil )
1814             @akapit_transzy ||= akapit_transzy
1815             @akat_walencyjny = akat_walencyjny?
1816             Ramka.odsmiec
1817             zrob_ramke_dup
1818             if session[:weryfikacja] and session[:weryfikacja][:werdykt] ==
                -1
1819                 @weryfikacja = session[:weryfikacja]
1820                 @werdykt = @weryfikacja[:werdykt]
1821             end
1823             if with_render # bo może być tylko jeden render w akcji.
1824                 render :update do |page|
1825                     page.replace_html( 'walencja', :partial => 'walencja' )
1826                 end
1827             end
1828         end# of refresh_walencje
1829     end # of if Walencja.x

1831     def refresh_prosby_o_nowy_wzorzec
1832         render :update do |page|
1833             page.replace_html( 'prosby_o_nowy_wzorzec',
1834                               :partial => 'prosby_o_nowy_wzorzec' )

```

```

1835     end
1836 end# of refresh_prosby_o_nowy_wzorzec

1839 def zrob_ramke_dup
1840   srd = session[:ramka_dup]
1841   if srd # w tym razie robimy ramkę do zrenderowania w części „ramki ze
        słownika”
1842     rf = Ramka.find( srd[:ramka_id] )
1843     @ramka_dup = [rf.ramka, srd[:ramka_id], srd[:elipsy],
        srd[:frazyluzne],
1844                 rf.orig_id, srd[:senczas_ok], srd[:zdduplicates],
        srd[:sloduplicates]
1845     ]

1847     ##      session[:ramka_dup] = {
1848     ##      :ramka_id => slويد, :elipsy => elipsy, :frazyluzne =>
        frazyluzne,
1851     ##      :sloduplicates => slويدuplicates, :zdduplicates =>
        zdduplicates }
1853     #
1854     # ram, ramka_id, elipsy, frazyluzne, orid, senczas_ok,
1855     ##      zdduplicates, slويدuplicates = ram
1857   end
1858 end # of zrob_ramke_dup

1861 def raiseifzero( obj, message )
1862   raise message if ( not obj ) or obj == 0
1863 end # of raiseifzero

1866 def prosby_toggle( które, val )
1867   h = { :odebrania_bp_pokaz => nil,
        :nowa_segmentacja_pokaz => nil}
1868   h[ które ] = val
1869   h.each_pair{ |k, v| session[ k ] = v }
1870 end

1874 def refresh_prosby( czy_liste_transz=nil )
1875   render :update do |page|
1876     ##      page.replace_html( 'flash_notice', :partial =>
        '/layouts/notice' )
1878     page.replace_html( 'prosby_annotatorek', :partial =>
        'prosby_annotatorek' )
1879     if czy_liste_transz
1880       page.replace_html( 'wsze_transze_otwarte', :partial =>
        'wsze_transze_otwarte' )
1881     end
1882   end # of render do |page|
1883 end# of refresh_prosby

1885 def render_prosby
1886   render :partial => 'prosby_annotatorek'
1887 end# of render_prosby

1891 def komentarz_toggle( val )

```

```

1892     session[ :komentarz_dodaj ] = val
1893     render :update do |page|
1894       page.replace_html(
1895         'komentarz_dodaj',
1896         :partial => 'komentarz_dodaj',
1897         :object => [AkapiTranszy.find(params[:id]),
1898                   params[:odrzuc]]
1898       )
1899     end
1900   end
1901
1902   end # of class
1903
1904   ## # Local Variables:
1905   ## # mode: ruby
1906   ## # End:

```

## application.rb

```

23   # Filters added to this controller apply to all controllers in the application.
24   # Likewise, all the methods added will be available for all controllers.
25
26   require 'jcode'
27   require 'vendor/plugins/gm/gm_methods'
28
29   # Enough is enough: I have enough the "Expected corpus_path.rb to define
30   # CorpusPath" error
31   ## load 'corpus_path.rb'
32   ## load 'frazatyp.rb'
33   ## load 'leksem.rb'
34   ## load 'rozbieznosc.rb'
35   ## load 'token_rozbiezny.rb'
36   ## load 'czasownik_auto.rb'
37   ## load 'interpretacja_anot.rb'
38   ## load 'odebrania_transz.rb'
39   ## load 'sens_anot.rb'
40   ## load 'transza_auto.rb'
41   ## load 'cz_m.rb'
42   ## load 'interpretacja.rb'
43   ## load 'plik_morph.rb'
44   ## load 'sensy_leksemu.rb'
45   ## load 'uzytkownik.rb'
46   ## load 'frazatyp.rb'
47   ## load 'klasa_gram.rb'
48   ## load 'protokol_rozbieznosci.rb'
49   ## load 'sensy.rb'
50   ## load 'akapit_transzy.rb'
51   ## load 'frazarozbiezna.rb'
52   ## load 'komentarz.rb'
53   ## load 'rola.rb'
54   ## load 'token.rb'
55   ## load 'akapit.rb'

```

```

81  # dalsze load'y przeniesione za klasę
83  class ApplicationController < ActionController::Base
84    # Pick a unique cookie name to distinguish our session data from others'
85    session :session_key => '_anotatornia_session_id'
87    filter_parameter_logging :haslo # , :uzid, :uzytkownik
88    # dla bezpieczeństwa: AWDWR s. 610
89    # 2008/07/14 jednak odślaniam :uzid i :uzytkownik, żeby było wiadomo,
    kto się logował.
90    include SslRequirement
92    include ApplicationHelper
93    # to chyba nie tak miało być, ale potrzebuję metod.
96    private
97    # nie-prywatne mogłyby być wywołane z przeglądarki.
99    def authorize
100      logger.info "##### tu działa #{session[:login].inspect}
    ({Rola.nazwa_rol(session[:rola_id])})"
101      unless Uzytkownik.find_by_uzytkownik_id(session[:uzid])
102        ##          session[:original_uri] = request.request_uri
    # nie będziemy pamiętać uri, bo niektóre podstrony wymagają
104        parametrów,
    # a bez parametrów dają błąd.
105        flash[:notice] = "Najpierw, proszę, zaloguj się."
106        redirect_to(:controller => "logowanie" , :action => "logowanie" )
107        return false
108      end
109    end
110  end
113  def check_role (* rola)
114    uzytkownik=Uzytkownik.find(session[:uzid])
115    if (not uzytkownik) or (not Rola.ktoras_z_rol?(uzytkownik, *
    rola) )
116      # jak widać, rola ma być symbolem roli do sprawdzenia
117      ##          session[:original_uri] = nil
118      flash[:notice] = "Nie masz uprawnień do wykonania tej czynności.
    Za karę wylogowuję Cię."
119      redirect_to(:controller => "logowanie" , :action =>
    "wylogowanie" )
120      return false
121    end
122  end
123  end
126  def check_admin
127    check_role(:admin)
128  end
131  def check_anotator
132    check_role(:anotator)
133  end
136  def check_annotant

```

```
137     check_role(:anotator, :zarzadca)
138   end
141   def check_annotant_ou_auditeur
142     check_role(:anotator, :zarzadca, :audytor)
143   end
145   def check_voyeur
146     check_role(:anotator, :zarzadca, :audytor, :gosc)
147   end
149   def check_zarzadca
150     check_role(:zarzadca)
151   end
153   def check_wszewid
154     check_role(:zarzadca, :audytor)
155   end
157   def check_audytor
158     check_role(:audytor)
159   end
162   def check_idid
163     if session[:uzid] != params[:id].to_i
164       session[:original_uri] = nil
165       flash[:notice] = "Próbowałeś/aś oglądać nie swój zasób.
166       Wylogowanie."
167       redirect_to(:controller => "logowanie" , :action =>
168         "wylogowanie" )
169       return false
170     end
171   end
172   def check_edytowalny
173     # jeśli pojawia się błąd w tej akcji, to sprawdź, czy nie jest wołana z
174     # dziwnego miejsca,
175     # np. bez session[:akapit].
176     rola_OK = ( Rola.anotator?(session[:rola_id]) ||
177       Rola.zarzadca?(session[:rola_id])) &&
178       ( ( not session[:drag_queen_id] ) || AnoVersion.port == 8004 )
179     if session[:akat]
180       akat = AkapitTranszy.find( session[:akat] )
181     elsif params[:akat]
182       akat = AkapitTranszy.find( params[:akat] )
183     else
184       logger.info( "#### brak session[:akat] i params[:akat]
185       #{Time.now.to_s(:db)}" )
186       akat = AkapitTranszy.znajdz(
187         session[:akapit],
188         (session[:anotator] || session[:uzid]))
189     end
190     logger.info "#### akapit #{akat.akapit_id} (tr.
191     #{akat.transza_id}, atr. #{akat.id})"
```

```

188   if DlaEli.nk : poziomy = params[ :poziom ]
189     raise "w wersji NKJP musi być podany poziom, np. :any!"
        unless poziomy
190       poziomy = poziomy.split('/').collect{ |poz| poz.intern }
191       poziomy = nil if poziomy == [ :any ]
192     else poziomy = nil
193   end
194   if poziomy
195     edytowalny = ( poziomy.collect{ |poz|
196                   akat.edytowalny_przez_annotatora( poz ) }.uniq ==
197                   [ true ] )
198     edytowalny &&=
199       (( akat.poziomy_dostepne( session[:rola_id],
200                                :jak_annotator => session[ :jak_annotator ]
201                                ) &
202        poziomy ) == poziomy )
203     ##      logger.info( "w wersji NKJP musi być podany poziom, np. :any!"
204     ##      edytowalny: #{edytowalny}" )
205   else # not poziom
206     edytowalny = akat.ma_Status?( :<, :zweryfikowany )
207     session[:status] = akat.status unless edytowalny
208   end
209   uzid_OK = ( akat.get_uzid == session[:uzid] )
210   if rola_OK and edytowalny and uzid_OK
211     return true
212   elsif Rola.zarzadca?(session[:rola_id])
213     # na wyraźne życzenie Eli zarządca może też zmieniać anotacje zdań
214     # zweryfikowanych.
215     if poziomy : return true # edytowalny poziom jest brany pod uwagę
216     # tylko w wersji NKJP, w wersji dla Eli jest nil.
217   else return true
218   end
219   ##      elsif Rola.gosc?(session[:rola_id])
220   ##      redirect_to( :controller=> 'anotacja', :action =>
221   ##      'ogladactwo' )
222   ##      return false
223   else
224     ##      logger.info( "check edytowalny #{Time.now.to_s(:db)}" +
225     ##      " rola_ok: #{rola_OK}, zakonczony: #{zakonczony},
226     ##      uzid_OK: #{uzid_OK}" )
227     ##      logger.info( "session:#{session[:uzid]},
228     ##      akat.uzid:#{akat.get_uzid}" )
229     if session[ :drag_queen_id ]
230       flash[ :notice ] = "Niestety, nie możesz anotować w dragu."
231     else
232       flash[ :notice ]="Akapt zweryfikowany lub nie masz prawa
233       zmienić anotacji."
234     end
235     redirect_to( :controller=> annotation_controller, :action =>
236     'anotuj', :akat => akat )

```

```
235     return false
236   end
237 end # of check_edytowalny

240 def check_drag_queen
241   # tak naprawdę używana tylko do zablokowania pobrania transzy
242   if session[:drag_queen_id]
243     flash[:notice] = "Niestety, nie możesz tego zrobić w dragu."
244     return false
245   else
246     return true
247   end
248 end # of check_drag_queen

251 def update_activity_time
252   if UNatrora.x :      timeout = 1600.minutes.from_now
253   else      timeout = 60.minutes.from_now
254   end
255   if session[:expires_at] and session[:expires_at] > Time.now
256     session[:expires_at]=timeout
257     return true
258   elsif session[:expires_at]
259     reset_session
260     flash[:notice] = "Sesja wygasła. Zaloguj się."
261     redirect_to :controller => 'logowanie', :action => "logowanie"
262     return false
263   else # nie ma :expires_at
264     session[:expires_at]=timeout
265     return true
266   end # of session[:expires_at]... i elsif
267 end # of update_activity_time

270 def przetwarzam_nie_przeszkadzac
271   # Chcielibyśmy uodpornić aplikację na wielokrotne naciskanie
    przycisku.
272   # i tę metodę wywołamy jako before_filter, a następną jako
    after_filter.
273   if session[:wlasnie_przetwarzam]
274     ##      flash[:notice] = "Ooops, jakiś błądzik, ale żyję i
        działałam."
276     session[:wlasnie_przetwarzam]=nil
277     if r_rola?( :audytor, :anotator, :gosc, :zarzadca )
278       ##      redirect_to :controller => :anotacja , :action =>
        :ogladactwo
280     else ##      redirect_to :controller => :logowanie, :action => :index
282     end
283     return true ## false
285   else # czyli nie ma :wlasnie_przetwarzam w sesji
286     session[:wlasnie_przetwarzam] = true
287     return true
288   end
```

```

289     end
290     # Kiedy naciskam np. Zatwierdź w widoku anotacji akapitu,
291     # nie jestem przekierowany, ale też się nie wyklada.

294     def skonczylam_przetwarzac
295         session[:wlasnie_przetwarzam] = nil
296         return true
297     end

300 end # of class.

304 # Z kontrolerami jeszcze poczekajmy, nie zdarzyło się "Expected..." z
    kontolerem.
305 # A jednak, także na nie przyszła pora
306 ## load 'admin_controller.rb'
308 ## load 'logowanie_controller.rb'
310 ## load 'anotacja_controller.rb'
312 ## load 'zarzadca_controller.rb'

315 ## # Local Variables:
316 ## # mode: ruby
317 ## # End:

```

### aux\_controller.rb

```

23 class AuxController < ApplicationController
24     layout 'admin'

26     before_filter :authorize
27     before_filter :update_activity_time

29     def sensy
30         @nounsenses = senses_arr( 'noun' )
31         @verbsenses = senses_arr( 'verb' )
32     end

35     private

37     def senses_arr( ksem )
38         [['lemat', 'sens_opis', 'anot_ct', 'zweryf_ct'].
39         collect{|hd| "<b>#{hd}</b>"}] +
40         SensySensemu.find(:all, :conditions => {:klasa_sem => ksem } ).
41         collect{|ss|
42             if ss.nowy?
43                 nowy = " (n)"
44             else nowy = ""
45             end
46             [ss.lemat, ss.sens_opis,
47              "#{ss.sens_anot_count}#{nowy}",
48              "#{ss.sens_zweryf_count}#{nowy}"]
49         }.sort
50     end # of senses_arr

53 end

```



**debug\_controller.rb**

```

23   require 'ramkowanie'

25   class DebugController < ApplicationController
26     layout 'admin'

28     before_filter :authorize
29     before_filter :update_activity_time

31     auto_complete_for :interpretacja, :reszta_tagu

34     def debug
35     end

37     def create_dummies
38       if AnoVersion.port == 8004
39         Dummy1.create!( :tresc => "rekord utworzony #{Time.now}" )
40         Dummy2.create!( :tresc => "rekord utworzony #{Time.now}" )
41       end

43       redirect_to :action => :debug
44     end

47     def elektroforeza
48       @to_sa_akapity = Zmienna4RadioButton.new
49       @to_sa_akapity.wartość = true
50       @maja_być_transze = Zmienna4RadioButton.new
51       @maja_być_transze.wartość = true
52     end

54     def znajdź_te_idy
55       @lista211 = ( params[:lista_idow] && params[ :lista_idow ][ :lista
56         ] )
57       @between211 = false
58       if @lista211 and @lista211 =~ /\^ * (\d+ * , * ) * (\d+)+$/ or
59         ( @between211 = ( @lista211 =~ /\^ * between +\d+ +and +\d+ * /i
60           ) )
61         @to_sa_akapity = params[ :to_sa_akapity ][ :wartość ].to_bool
62         @maja_być_transze = params[ :maja_być_transze ][ :wartość
63           ].to_bool

64         render :update do |page|
65           if @maja_być_transze
66             page.replace_html "lista_transz",
67               :partial => "lista_transz"
68           else
69             page.replace_html "lista_ścieżek",
70               :partial => "lista_sciezek"
71           end # of transze or ścieżki
72         end # of render

73     else

```



```
63         else
64             flash[ :notice ] = "Nie ma <b>anotatora</b> #{logins[-1]},
                                   loguję Cię jako #{login}."
65         end
66     else
67         session[ :login ] = login
68     end

71     logger.info "=== #{session[:login]} logged in at
                  #{Time.now.to_s(:db)}"
72     # uri = session[:original_uri]
73     # session[:original_uri] = nil
74     if r_anotator?
75         redirect_to :controller => annotation_controller,
76                     # application helper
77                     :action => 'lista_transz'
78     elsif Rola.wszewid?(session[:rola_id]) or
79         Rola.gosc?(session[:rola_id])
80         redirect_to :controller => annotation_controller, :action =>
81             'ogladactwo'
82     else redirect_to( # uri ||
83                     { :action => "index" })
84     end

85     elsif uzytkownik == :zablokowany
86         flash[:notice] = "Twoje konto zostało zablokowane. <br>
87                             Skontaktuj się z administratorem."
88     else
89         flash[:notice] = "Nietrafna kombinacja login/hasło"
90     end
91 end # of logowanie

92 def wylogowanie
93     # session[:uzid] = nil
94     # session[:rola_id] = nil
95     logger.info "=== #{Uzytkownik.find(session[:uzid]).login} logged
96                 out at #{Time.now.to_s(:db)}"
97     reset_session
98     if flash[:notice]
99         flash[:notice]=flash[:notice]
100     else
101         flash[:notice] = "Wylogowałeś/aś się"
102     end
103     redirect_to(:action => "logowanie" )
104 end

105 def index
106 end

109 def session_expiry
110     @time_left = (session[:expires_at] - Time.now).to_i
```

```

111     if @time_left < 1.minute
112       render :update do |page|
113         page.replace_html 'expiry', :partial => 'session_expiry',
114           :object => @time_left
115       end
116     end
117     unless @time_left>0
118       flash[:notice] = "Twoja sesja wygasła. Proszę, zaloguj się."
119     # render do |page|
120     #   page « "window.location = 'url_for(:controller => "logowanie" ,
121     #     :action => "wylogowanie" );"
122     # end
123     # render '/logowanie/redirect'
124     reset_session
125     # redirect_to(:controller => "logowanie" , :action => "wylogowanie" )
126   end
127 end
128 end
129 end
130 end

```

### superanotacja\_controller.rb

```

23   Uzytkownik.find( :first ) # żeby utworzyć Natrorów, gdyby ich nie było.
24
25   class SuperanotacjaController < AnotacjaController
26
27     before_filter :check_zarzadca
28
29     ##   skip_before_filter :check_edytowalny, :only => [
30       :skomentowane_tylko_nowe_przełącz,
31       ##
32       :skomentowane_przełącz]
33     ##   wygląda na to, że zastępuje klauzulę # :except filtra klasy
34     ##   rodzicielskiej. Więc robię rzecz mocno nieelegancką: dopisuję symbole tych
35     ##   metod w klasie rodzicielskiej.
36
37     before_filter :check_edytowalny, :except => EDYTOWALNY_EXCEPTIONS
38     + [
39
40       :skomentowane_tylko_nowe_przełącz,
41       :skomentowane_tylko_nowe_przełącz,
42       :skomentowane_przełącz,
43       :koment-
44       arz_nowy_przełącz,
45       :odbierz_transze_bprosb
46       :odebrania_bp_ukryj,
47       :odebra-
48       nia_bp_pokaz
49     ]
50
51     def nowa_dyzambiguacja
52       set_uzid
53       intid=params[ :interpretacja_anot ]
54
55       if intid

```

```

50     intid=intid[:interpretacja_id].to_i
51     tok = Interpretacja.find( intid ).token
52     @tokid = tok.id
53     @interpretacja_annot=InterpretacjaAnot.find(
54         :first,
55         :conditions => {
56             :akapit_transzy_id =>
57                 akapit_transzy.id, # to jest
58                 dobrze, to jest akat
59                 przekazany w parametrach.
60                 :token_id => @tokid
61             })
62     @interpretacja_annot ||= InterpretacjaAnot.new
63     @interpretacja_annot.interpretacja_id = intid
64     @interpretacja_annot.uzytkownik_id=@uzid
65     @interpretacja_annot.akapit_transzy_id = akapit_transzy.id
66     @interpretacja_annot.token_id = @tokid
67     InterpretacjaAnot.transaction do
68         @interpretacja_annot.save!
69         @interpretacja_annot.skopiuj_na_bliźniaczą # nyhet
70         if DlaEli.nk
71             obniz_status( akapit_transzy, :morphosyntactic)
72             tok.odsmiec_interpretacje
73         else
74             obniz_status( akapit_transzy )
75         end
76     end
77     else# not intid
78         @tokid=params[:id].to_i
79     end
80     # redirect_to :action => 'anotuj' for debug
81     raiseifzero( @tokid, "nowa_dyzambiguacja" )
82     if ( szd = session[ :zmieniaj_dyzamb ] ) and szd[ @tokid ]
83         szd[ @tokid ] = nil
84     end
85     zdeweryfikuj( :morphosyntactic, @tokid )
86     odswiez_bitoken( @tokid )
87     end # of nowa_dyzambiguacja.
88
89     def morfoskładnia_zatwierdź_tę
90         set_uzid
91
92         tok = Token.find( params[:id] )
93         @tokid = tok.id
94         @ia=InterpretacjaAnot.find(
95             :first,
96             :conditions => {
97                 :akapit_transzy_id =>
98                     akapit_transzy.id,
99                     :token_id => @tokid
100             })

```

```

98      @ia ||= InterpretacjaAnot.new
99      changée = @ia.new_record?
100     dis_id = params[ :disamb_id ].to_i
101     if @ia.interpretacja_id != dis_id
102       @ia.interpretacja_id = dis_id
103       changée = true
104     end
105     if changée
106       @ia.uzytownik_id=@uzid
107       @ia.akapit_transzy_id = akapit_transzy.id
108       @ia.token_id = @tokid
109     end
110     InterpretacjaAnot.transaction do
111       @ia.save! if changée
112       @ia.skopiuj_na_bliźniaczę
113       obniz_status( akapit_transzy, :morphosyntactic)
114       tok.odsmiec_interpretacje
115     end # of transaction

117     raiseifzero( @tokid, "nowa_dyzambiguacja" )
118     if ( szd = session[ :zmieniaj_dyzamb ] ) and szd[ @tokid ]
119       szd[ @tokid ] = nil
120     end
121     zdeweryfikuj( :morphosyntactic, @tokid )
122     odswiez_bitoken( @tokid )
123   end # of morfoskładnia_zatwierdź_tę.

126   def self.zaślepka( meth )
127     eval "def #{meth}raise \"zaślepka\"end"
128   end # of zaślepka

131   def wybor_sensu # wołana z _token_wybs.html
132     set_uzid
133     akat_hash # inicjuje @akat_hash, jeśli jej nie było
134     obniz_status( @akapit_transzy, :word_senses )
135     sensid = params[ :the_sens ]
136     tokid = params[ :id ].to_i
137     disaid = params[ :disamb_id ].to_i
138     raiseifzero( tokid, "wybor_sensu")

140     if sensid and sensid[:sensy_id] and
141       sensid[:sensy_id].to_i != 0
142       # drugi człon koniunkcji na wypadek,
143       # gdyby nie nie wybrano, ale to i tak mało
144       sensid=sensid[ :sensy_id ].to_i

146     [ @akapit_transzy, @akat_bliźniak ].each { |at190|
147       @sens_anot = SensAnot.znajdz( tokid, at190.akat_hash )
148       @sens_anot ||= SensAnot.new
149       @sens_anot.sensy_id = sensid
150       @sens_anot.uzytownik_id = @uzid
151       @sens_anot.akapit_transzy_id = at190.id

```

```

152         @sens_anot.interpretacja_id = disaid
153         @sens_anot.token_id = tokid
154         @sens_anot.automatycznie = false
155         s1187 = @sens_anot.sensy( :refresh )
156         if s1187
157             @sens_anot.save!
158         else
159             @sens_anot = nil
160         end
161     }
162     end # of sensid and sensid[:sensy_id] and sensid[:sensy_id].to_i != 0
163     zdeweryfikuj( :word_senses, tokid )
164     odswiez_bitoken(tokid, :tylko_sens)
165 end # of wybor_sensu

168     zaślepka :fraza_upuszczono

171 def frazy_wyczysc
172     akat = akapit_transzy
173     AkapitTranszy.transaction do
174         obniz_status( akat )
175         akat.fraza_anot.delete_all
176         @akat_bliźniak.fraza_anot.delete_all # nyhet
177     end

179     redirect_to :action => :anotuj, :akat => akat, :anchor =>
        :primafalsa
180 end # of frazy_wyczysc

182     zaślepka :fraza_wyczysc
183     zaślepka :fraza_ukryj_typ
184     zaślepka :fraza_wybor_typu
185     zaślepka :fraza_glowy_ukryj
186     zaślepka :fraza_synth_ukryj
187     zaślepka :fraza_synth_ukryj
188     zaślepka :fraza_dwie_glowy
189     zaślepka :fraza_wybor_glowy
190     zaślepka :fraza_sie_typ_ukryj
191     zaślepka :fraza_sie_typ_wybor

194 def interpretacja_z_tagu
195     set_uzid
196     @tokid = params[:id].to_i
197     tagparam = "token_tag#{@tokid}".intern
198     lematparam="token_lemat#{@tokid}".intern
199     tag = params[tagparam][:tag]
200     lemat = params[lematparam][:lemat]
201     ct = Tagset.check_tag( tag, lemat )
202     if ct[0]
203         iid = Interpretacja.fc_by_fulltag( @tokid, lemat, tag ).id
204         conds = {
205             :akapit_transzy_id => akapit_transzy.id,

```

```

206         :token_id => @tokid
207     }
208     ia = InterpretacjaAnot.find( :first, :conditions => conds )
209     ia ||= InterpretacjaAnot.new( conds )
210     ia.uzytkownik_id = @uzid
211     ia.interpretacja_id = iid
212     ia.save!
213     ia.skopiuj_na_bliźniaczą # nyhet
214     Interpretacja.odsmiec( :token_id => @tokid )

216     if ( szd = session[ :zmieniaj_dyzamb ] ) and szd[ @tokid ]
217         szd[@tokid] = nil
218     end
219     zdeweryfikuj( :morphosyntactic, @tokid )
220     odswiez_bitoken( @tokid )
221     else flash.now[:tworz_tag] = ct[1]
222         session[ :tworz_tag ] ||= Hash.new
223         session[ :tworz_tag ][ @tokid ] = { :lemat => lemat, :tag =>
            tag }
224         odswiez_token( @tokid, :tworz_tag )
225     end
226 end# of interpretacja_z_tagu.

228 # nowytag_skopiuj_orth jest O.K. w oryginale

231 def segmentacja_ujednoznacznij
232     set_pozioomy
233     segdec=params[:sg_variant_annot]
234     ##      logger.info( "### segdec: #{segdec}, class:
        #{segdec.class}" )
236     session[:zmieniaj_segmentacje] ||= Hash.new
237     if segdec
238         segdec=segdec[ :wybierz ].to_bool
239         tokid = params[:id].to_i
240         token = Token.find( tokid )
241         sg_choice = token.sg_choice
242         sgvid = token.sg_variant_id
243         sguj_tokids = []
244         SgVariantAnot.transaction do
245             sg_choice.sg_variant.each{ |sgv|
246                 sgv_cond = {
247                     :akapit_transzy_id => akapit_transzy.id,
248                     :sg_choice_id => sg_choice.id,
249                     :sg_variant_id => sgv.id
250                 }
251                 sgvan = SgVariantAnot.find(
252                     :first,
253                     :conditions => sgv_cond)
254                 sgvan ||= SgVariantAnot.new( sgv_cond )
255                 sgvan.chosen = (( sgv.id == sgvid ) == ( segdec == true ))
256                 sgvan.save!

```



```

257         sgvan.skopiuuj_na_bliźniaczy # nyhet
258         sgv.token.each{ |tok| sguj_tokids « tok.id }
259     }# of each segm. variant
260     obniz_status( akapit_transzy, :segmentation )
261     end# of transaction
262 else# not intid
263     sguj_tokids = [ params[:id].to_i ]
264 end
265 #     redirect_to :action => 'anotuj' for debug
266 sguj_tokids.each{ |tokid|
267     raiseifzero( tokid, "segmentacja_ujednoznaczniej" )
268     session[ :zmieniaj_segmentacje ][ tokid ] = nil }

270     zdeweryfikuj( :segmentation, sguj_tokids )
271     odswiez_bitoken( sguj_tokids, :segmentation )

273 end # of segmentacja_ujednoznaczniej.

275 def segmentacja_zatwierdź_tę
276     set_poziomy
277     tokid = params[:id].to_i
278     token = Token.find( tokid )
279     sg_choice = token.sg_choice
280     sguj_tokids = []
281     SgVariantAnot.transaction do
282         sg_choice.sg_variant.each{ |sgv|
283             sgv_cond = {
284                 :akapit_transzy_id => akapit_transzy.id,
285                 :sg_choice_id => sg_choice.id,
286                 :sg_variant_id => sgv.id
287             }
288             sgvan = SgVariantAnot.find(
289                 :first,
290                 :conditions => sgv_cond)

292             sgvan.skopiuuj_na_bliźniaczy # nyhet
293             sgv.token.each{ |tok| sguj_tokids « tok.id }
294         }# of each segm. variant
295         obniz_status( akapit_transzy, :segmentation )
296     end# of transaction

298     sguj_tokids.each{ |tokid|
299         raiseifzero( tokid, "segmentacja_ujednoznaczniej" )
300         if session[ :zmieniaj_segmentacje ]
301             session[ :zmieniaj_segmentacje ][ tokid ] = nil
302         end
303     }

305     zdeweryfikuj( :segmentation, sguj_tokids )
306     odswiez_bitoken( sguj_tokids, :segmentation )

308 end # of segmentacja_zatwierdź_tę.

311 def konczymy_zdanie

```

```

312     @tokid=params[ :id ].to_i
313     set_poziomy
314     akat=akapit_transzy
315     kz = KoniecZdaniaAnot.znajdz( akat, @tokid )

317     if kz
318         kz.zniszcz_oba
319     else
320         KoniecZdaniaAnot.zrób_oba(
321             :akat => akat,
322             :uzytkownik_id => @uzid,
323             :token_id => @tokid
324         )
325     end
326     zdeweryfikuj( :sentences, @tokid )
327     odswiez_bitoken( @tokid, :sentences )
328 end # of konczymy_zdanie.

331 def word_senses_zatwierdź_tę
332     set_poziomy
333     tokid = params[:id].to_i
334     akat190 = AkapitTranszy.find( params[ :akat ].to_i )
335     akath190 = akat190.akat_hash
336     tok190 = Token.find( tokid )
337     the_sens190 = tok190.the_sens( akath190 ).id
338     tsa190 = tok190.sens_anot
339     SensAnot.transaction do
340         tsa190.each{ |sa190|
341             unless sa190.sensy_id == the_sens190
342                 sa190.sensy_id = the_sens190
343                 sa190.save!
344             end}

346         if tsa190.size < 2
347             sa190 = SensAnot.create!(
348                 :sensy_id => the_sens190,
349                 :uzytkownik_id => @uzid,
350                 :akapit_transzy_id => akat190.blizniaczy_id,
351                 :interpretacja_id => tok190.do_sensu?( akath190 ),
352                 :token_id => tok190.id,
353                 :automatycznie => false )
354         end

356         obniz_status( akat190, :word_senses )
357     end# of transaction

359     zdeweryfikuj( :word_senses, tok190.id )
360     odswiez_bitoken( tok190.id, :tylko_sens )

362 end # of word_senses_zatwierdź_tę.

365 def skomentowane_przełącz
366     session[:skomentowane_pokaż] = (not session[:skomentowane_pokaż])

```

```
367     if ptn = params[:tylko_nowe]
368       ustaw_tylko_nowe_komentarze_ogladactwo( ptn.to_bool )
369     end
370     render_skomentowane
371 end # of skomentowane_przełącz

374 def skomentowane_tylko_nowe_przełącz
375   ustaw_tylko_nowe_komentarze_ogladactwo(
376     (not session[:tylko_nowe_komentarze_ogladactwo])
377   )
378   render_skomentowane
379 end # of skomentowane_tylko_nowe_przełącz

381 def skomentowane_tylko_nowe_anotacja_przełącz
382   ustaw_tylko_nowe_komentarze_anotacja(
383     (not session[:tylko_nowe_komentarze_anotacja])
384   )
385   render_komentarze
386 end

388 def komentarz_nowy_przełącz
389   ko = Komentarz.find( params[:id] )
390   ko.nowy = ( not ko.nowy? )
391   ko.save!
392   render :update do |page|
393     page.replace( "komentarz_ballot-#{ko.id}",
394       :partial => 'komentarz_ballot', :object => ko
395     )
396   end # of render
397 end # of komentarz_nowy_przełącz

399 # spełnij_prośbę i odrzuć_prośbę przeniesione do AnotacjaController, by
    miał do nich dostęp audytor.

402 def odbierz_transze_bprosby
403   ti = params[:transza_odbierana][:numer].to_i
404   if t = Transza.find_by_transza_id(ti)
405     ui=t.uzytownik_id
406   end
407   if ti and ui
408     params[:id] = ProsbyAnotatorek.create(
409       :dotyczy_id => ti,
410       :dotyczy_type => "Transza",
411       :uzytownik_id => ui,
412       :opis => "odebrana z inicjatywy Zarządcy"
413     ).id
414   end
415   refresh_prośby
416 end # of odbierz_transze_bprosby
```

```

419 def odebrania_bp_ukryj
420   prośby_toggle( :odebrania_bp_pokaz, nil )
421   render_prośby
422 end

424 def odebrania_bp_pokaz
425   prośby_toggle( :odebrania_bp_pokaz, true )
426   render_prośby
427 end

430 def zmien_strony_akapitow
431   ##      raise "obsolete method called: #{annotation_controller.capitalize}Controller.zmien_strony_akapitow"
432   # mamy na nią filter check_zaradca i tylko po to robimy osobną akcję
433   raise "zmień_strony_akapitów: no akat given???" unless params[:akat]
434   raise "zmień_strony_akapitów: no akat given???" unless params[:akat]
435   redirect_to :action => :ustal_akapit,
436   :anotator => params[:id],
437   :akat => params[:akat]
438 end

440 def sensowanie_anulowane
441   t187 = Token.find( params[:id] )
442   odswiez_bitoken( t187.id, :tylko_sens )
443 end

447 private

449 def render_skomentowane
450   render :update do |page|
451     page.replace_html(
452       'akapity_skomentowane',
453       :partial => 'akapity_skomentowane'
454     )
455   end # of render
456 end # of render_skomentowane

459 def render_komentarze
460   # (w widoku anotuj)
461   akapit_transzy
462   render :update do |page|
463     page.replace_html(
464       'komentarze',
465       :partial => 'komentarze', :object => [
466         @akapit_transzy, :anotacja ]
467     )
468   end # of render
469 end # of render_komentarze

473 def zdeweryfikuj( poziom, tokids )
474   # metoda, która usunie id tokenu z hasza weryfikacji, by się już nie
475   wyświetlało na żółto.
476   logger.info "=== weryfikacja przed:
477     #{session[:weryfikacja].inspect}"

```

```

476     if (w = session[:weryfikacja] ) and w[ poziom ]
477       w[poziom] -= ( [ tokids ].flatten )
478     end
479     [ Token.find( tokids ) ].flatten.each{ |tok|
480       tok.superancja ||= Array.new
481       ##       logger.info "=== tok.superancja ==
482       #{tok.superancja.inspect}"
483       tok.superancja |= [ poziom ]
484       tok.save!
485       tok.zambiguuj if poziom == :morphosyntactic
486     }

487     if poziom == :segmentation
488       akapit_transzy.akapit.token_rejected.each{ |tok|
489         tok.chosen = true
490         tok.save!
491       }
492     end
493   end

494   unless akapit_transzy.ma_status?( poziom, :==, :osadzany ) and
495     @akat_bliźniak.ma_status?( poziom, :==, :osadzany )
496     # próba posłania != dawała wyjątek „undefined method ‘!’ for
497     o:Fixnum”.
498     akapit_transzy.set_status( poziom, :osadzany )
499     ##       @akat_bliźniak.set_status( poziom, :osadzany,
500     :tylko_ten )
501   end

502   logger.info "==== weryfikacja po:
503   #{session[:weryfikacja].inspect}"
504 end # of zdeweryfikuj

505 def odswiez_bitoken( * params )
506   # różni się od odswiez_token tym, że odświeża element po obu stronach.
507   tokids = params.delete_at(0)
508   logger.info "==== tokids w odśwież_bitoken:
509   #{[tokids].flatten.join(', ')}"
510   tokens = Token.find( :all,
511     :conditions => { :token_id => [tokids].flatten,
512     :chosen => true } # drugi warunek dodany
513     2009/8/23 w ramach tropienia #246.
514   ) # musi umieć obsłużyć tablicę idów, ze względu na
515     segmentacja_ujednoznaczni.

516   set_uzid
517   akat_hash
518   render :update do |page|
519     if not ( params & [ :tylko_sens, :segmentation, :sentences ] )[ 0
520     ]
521       if params.include?( :tworz_tag )
522         tokpartial = 'token_tworz_tag'
523       else
524         tokpartial = 'token_disamb'

```

```

523         session[:tworz_tag] = nil if session[:tworz_tag]
524     end
525     tokens.each{ |tok|
526         page.replace_html
527             "disa-token_#{@akapit_transzy.id}_#{tok.id}",
528             :partial => tokpartial, :object => [tok, @akapit_transzy,
529                 true]
530
531         page.replace_html
532             "disa-token_#{@akat_bliźniak.id}_#{tok.id}",
533             :partial => tokpartial, :object => [tok, @akat_bliźniak,
534                 true] # nyhet
535     }
536
537     elsif (params & [ :segmentation, :sentences ])[ 0 ]
538         tokens.each{ |tok|
539             page.replace
540                 "outer_seg-token_#{@akapit_transzy.id}_#{tok.id}",
541                 :partial => 'token_td', :object => [tok, @akapit_transzy,
542                     true]
543
544             page.replace
545                 "outer_seg-token_#{@akat_bliźniak.id}_#{tok.id}",
546                 :partial => 'token_td', :object => [tok, @akat_bliźniak,
547                     true] # nyhet
548         }
549     else
550         tokens.each{ |tok|
551             page.replace_html
552                 "sens-anot_#{@akapit_transzy.id}_#{tok.id}",
553                 :partial => 'token_sens', :object => [tok, @akapit_transzy,
554                     true]
555
556             page.replace_html "sens-anot_#{@akat_bliźniak.id}_#{tok.id}",
557                 :partial => 'token_sens', :object => [tok, @akat_bliźniak,
558                     true]
559         }
560     end
561
562     if params.include?( :sentences )
563         page.replace_html(
564             'sentenśnik',
565             :partial => 'sentenśnik', :object =>
566                 @akapit_transzy
567             # z jakiegoś powodu nie działało, kiedy wpisywałem tu
568             akapit_transzy
569         )
570     end
571
572     page.replace_html(
573         'status_info',
574         :partial => 'status_info'
575     )

```

```

562             # z jakiegoś powodu nie działało, kiedy wpisywałem tu
             akapit_transzy
563         )
564         page.replace_html(
565             'zatwod_buttons',
566             :partial => 'anotuj_zatwod_nk'
567             # z jakiegoś powodu nie działało, kiedy wpisywałem tu
             akapit_transzy
568         )

570         page.replace( 'blizniaczy_status', :partial =>
            'blizniaczy_status' )
571     end
572 end # of odswiez_bitoken.

574 # refresh_prośby zdefiniowane w anotacja_controller.rb

577 def ustaw_tylko_nowe_komentarze_ogladactwo( value )
578     u = Uzytkownik.find( session[ :uzid ] )
579     u.tylko_nowe_komentarze_ogladactwo = value
580     u.save!
581     session[ :tylko_nowe_komentarze_ogladactwo ] =
        u.tylko_nowe_komentarze_ogladactwo?
582 end

585 def ustaw_tylko_nowe_komentarze_anotacja( value )
586     u = Uzytkownik.find( session[ :uzid ] )
587     u.tylko_nowe_komentarze_anotacja = value
588     u.save!
589     session[ :tylko_nowe_komentarze_anotacja ] =
        u.tylko_nowe_komentarze_anotacja?
590 end

594 end # of class

```

### zarzadca\_controller.rb

```

23 class ZarzadcaController < ApplicationController
24
25     layout "admin"
26
27     before_filter :authorize
28     before_filter :update_activity_time, :except => :session_expiry
29     before_filter :check_zarzadca
30
31     def lista_anotatorow
32         @wszyscy_anotatorzy = Uzytkownik.find(
33             :all,
34             :order => :login,
35             :conditions => { :rola_id =>
36                 Rola.rid(:anotator) }
37         )
38     end
39
40     def odbierz_transze

```

```

41   @transza=TranszaAuto.find(params[:id])
42   @anotator=Uzytkownik.find(params[:anotator])
43   if request.post?
44     if @transza.uzytkownik_id == @anotator.uzytkownik_id
45       @transza.uzytkownik_id=nil
46       @transza.save
47       flash[:notice]= "Odebrałam transzę #{@transza.transza_auto_id}
48       " +
49       "anotatorowi #{@anotator.login}."
50     else
51       flash[:notice] = "Próba odebrania transzy
52       #{@transza.transza_auto_id} " +
53       "uzytkownikowi #{@anotator.login} nie powiodła się."
54     end
55     redirect_to (:action => "lista_anotatorow")
56   end
57 end
58 end

```

## app/helpers

### admin\_helper.rb

```

23 module AdminHelper
24   end

```

### anotacja\_helper.rb

```

23 module AnotacjaHelper
24
25   def akat_edytowalny?( poziom=nil, akat=nil, lewy=true )
26     if lewy or r_zarzadca?
27       uzid = session[ :uzid ]
28       akat ||= AkapitTranszy.find( session[ :akat ] || params[ :akat ]
29       )
30       if poziom
31         return akat.poziomy_dostepne(
32           session[ :rola_id ],
33           ##
34           :segmentation_specjalnie => true,
35           # zmienione na wyraźne życzenie AP
36           2009/7/28
37           :jak_anotator =>
38           session[:jak_anotator],
39           :poziomy_pokazne => session[ :poziomy
40           ] ).
41       end
42       include?( poziom )
43     else # not poziom — patrzemy na duży Status
44       status = ( session[ :status ] || akat.status )
45     end
46   end
47 end

```



```

42     if Rola.anoator?( session[ :rola_id ] )
43       wynik = (AkapitTranszy.editable_par_annotateur?( status ))
44       if session[ :anoator ]
45         return((session[:anoator] == uzid) and wynik)
46       else return wynik
47       end
48     elsif Rola.zarzadca?( session[:rola_id])
49       return true
50     else return false
51     end # of if Rola.anoator?

53   end# of if poziom

55   else # not lewy or super
56     return false
57   end # of if lewy

59 end# of akat_edytowalny?

61 # Ale dlaczego mamy test edytowalności w dwóch miejscach: tu i w
application.rb, check_edytowalny ?
62 # Ten jest taki wstępny, żeby coś wyświetlić lub nie.
63 # Ten w check_edytowalny wykonuje się przed konkretnymi działaniami.

65 def znak_zatwierdź_tę
66   "&#9745;"
67 end

70 def akat_proszalny_segmentnie?
71   akat ||= AkapitTranszy.find( session[ :akat ] || params[ :akat ] )
72   akat.ma_status?( :morphosyntactic, :<=, :po_poprawce )
73 end # of akat_proszalny_segmentnie?

77 end# of module

79 ## # Local Variables:
80 ## # mode: ruby
81 ## # End:

```

## application\_helper.rb

```

23 # Methods added to this helper will be available to all templates in the
application.
24 # włączam go także do kontrolera AnotacjaController.

26 module ApplicationHelper

28   def ajax_request(argum)
29     "new Ajax.Request('#{argum}', {asynchronous:true,
evalScripts:true,parameters:Form.serialize(this.form)}); return
false;"
30   end

32   def ajax_updater(argum1, argum2)

```

```
33      "new Ajax.Updater("#{argum1}", "#{argum2}", {asynchronous:true,
    evalScripts:true, parameters:Form.serialize(this.form)}); return
    false;"
34  end

36  def r_gosc?
37    r_rola?( :gosc )
38  end

41  def r_anotator?
42    r_rola?( :anotator )
43  end

46  def r_zarzadca?
47    r_rola?( :zarzadca )
48  end

50  def r_wszewid?
51    r_rola?( :zarzadca, :audytor )
52  end

55  def r_admin?
56    r_rola?( :admin )
57  end

60  def r_rola?( * role )
61    ##      logger.info(" uzid: #{session[:uzid]}")
63    Rola.ktoras_z_rol?( session[ :uzid ], * role )
64  end

66  def uz_login( uzid )
67    Uzytkownik.find( uzid ).login if uzid
68  end

70  def params_to_arri( key )
71    # konwersja parametru na tablicę integerów
72    params[key].to_s.split('/').collect{|x| x.to_i }
73  end

75  def annotation_controller
76    if r_zarzadca? : 'superanotacja'
77    else 'anotacja'
78    end
79  end

81  def log_time( log_text="" )
82    logger.info "### #{log_text}: " + Time.now.to_s( :db )
83  end

85  end# of module.

87  ## # Local Variables:
88  ## # mode: ruby
89  ## # End:
```

**debug\_helper.rb**

```

23 module DebugHelper
24   end

```

**logowanie\_helper.rb**

```

23 module LogowanieHelper
24   end

```

**migration\_helper.rb**

```

24   require 'jcode'
26   ## require 'connection_pool'

30 module MigrationHelper
31   # metody tego modułu zwracają true jeśli coś zrobią lub false, jeśli okażą
   się no-opem.

34   def addcolumn( tablename, colname, colspec )
35     unless tablename.to_s.camelize.constantize.column_names.include?(
       colname.to_s )
36       execute "alter table #{tablename} add column #{colname}
         #{colspec}"
37       return true
38     else return false
39     end
40   end # of addcolumn

43   def addindex( tablename, columns, unique=nil )
44     dtname = tablename.to_s.split('.')
45     tname = dtname.delete_at( -1 )
46     dbname_dot = if dtname[0]
47                   dtname[0] + '.'
48                 end

50     cols = [ columns ].flatten
51     indexname = ([tname] + cols).join('_')
52     if ActiveRecord::Base.find_by_sql( "pragma index_info(
       #{indexname} )" )[0]
53       return false
54     else
55       execute " create #{if unique : 'unique ' end}index if not exists
         " +
56         " #{dbname_dot}#{indexname} on #{tname}( #{cols.join(', ')});"
57       return true
58     end
59   end # of addindex

62   def dropindex( indexname )

```

```

63     if ActiveRecord::Base.find_by_sql( "pragma index_info(
        #{indexname} )" ) [0]
64         execute " drop index if exists #{indexname}; "
65         return true
66     else return false
67     end
68 end

71 def droptable( tablename, force=nil )
72     if tables.include?( "#{tablename}" ) or force # uwaga! tu jest
        zwykle include?, które odróżnia symbol od stringa, dlatego wziąłem
        w cudzysłów
73         execute " drop table if exists #{tablename}; "
74         return true
75     else return false
76     end
77 end

80 def przepisz_tabelę( tabledef, * indices )
81     tabledef =~ /\^[^()]* (?:\(\w+\)\.)* (\w+) * \(\/
82     dbname = $1
83     tname = $2
84     dtname = [$1, $2].compact.join('.')

86     oldcols = tname.camelize.constantize.column_names

88     puts tname.inspect

90     columns = []
91     tabledef.gsub( /(?:\(|,|-.* \n)\s* (\w+)/ ) { |match|
        # [database-name.]table-name
92         columns « $1
93     }

95     puts columns.inspect

97     columns &= oldcols
98     coljoin = columns.join(', ')
99     puts coljoin

101     execute " alter table #{dtname} rename to #{tname}_old;"
102     execute tabledef
103     execute " insert into #{dtname} (#{coljoin})select #{coljoin} from
        #{tname}_old "

105     droptable tname + "_old", :force
106     droptable dtname + "_old", :force

108     indices.each{ |ind|
109         if ind.kind_of?( String ) and ind =~ /\bcreate\b.* \bindex\b/i
110             execute ind
111         else
112             cols, unique = ind
113             addindex( dtname, cols, unique )
114         end

```

```

115     }
116   end # of przepisz_tabele

119   def pragmas_for_update
120     execute " pragma locking_mode = exclusive; "
121     execute " pragma synchronous=off; "
122   end

124   def pragmas_normal
125     execute " pragma locking_mode = normal; "
126     execute " pragma synchronous=normal; "
127   end

130   unless defined?( $anotatorni _killed )
131     $anotatorni _killed = false
132   end

135   def kill_anotatorni _if_8003
136     if AnoVersion.port == 8003
137       unless $anotatorni _killed
138         'killall anotatornia-keeprrunning'
139         './anotatornia-mongrels stop'
140         'tar -cjvf bazy-bk-\`date +%F-%H%M\`'.tar.bz2
141         anotatornia_8003.db uzytkownicy_8003.db' unless UNatrora.x
142         $anotatorni _killed = true
143       end
144     end
145   end

146   def kill_anotatorni _if_8003_nobackup
147     if AnoVersion.port == 8003
148       unless $anotatorni _killed
149         'killall anotatornia-keeprrunning'
150         './anotatornia-mongrels stop'
151       end
152     end
153   end

155 end # of module

159 class Uzytkownicy8003 < ActiveRecord::Base
160   self.abstract_class = true
161   establish_connection(
162     ## establish_connection (Act-
163     iveRecord::ConnectionAdapters::ConnectionHandler)
164     ## connection(
165     ActiveRecord::Base.establish_connection(
166       :adapter => "sqlite3",
167       :database => "uzytkownicy_8003.db" )
168   end

zarzadca_helper.rb

23 module ZarzadcaHelper
24   end

```

**zdanie\_helper.rb**

```
23 module ZdanieHelper
24   end
```

**app/models****akapit.rb**

```
1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: akapit
5   #
6   # akapit_id :integer primary key
7   # morph_id :integer not null
8   # path_id :integer not null
9   # segmentation_xmlid :text
10  # segmentation_xlink_href :text
11  # morphosyntactic_xmlid :text
12  # created_at :timestamp
13  # updated_at :timestamp
14  # tresc :text
15  # incipit :text
16  #
17
40  require 'ramkowanie'
41
42  class Akapit < ActiveRecord::Base
43
44    belongs_to :morph
45    has_many :token, :order => "kolejnosc", :conditions => { :chosen =>
      true }
46    has_many :token_all, :class_name => "Token", :order => "kolejnosc"
47
48    has_many :token_incipny, :class_name => "Token",
49      :select => "akapit_id, token_id, orth, ns_nastepuje, ns_poprzedza",
50      :conditions => { :chosen => true },
51      :order => "kolejnosc",
52      :limit => 11
53
54    # „Wyjście morfoskładni ma nie zawierać segmentów odrzuconych.
55    # Informacja o wszystkich wariantach segmentacyjnych jest tylko
56    # w ann_segmentation.xml
57    has_many :token_rejected, :class_name => "Token", :order =>
      :kolejnosc,
58    :conditions => { :chosen => false }
59
59    has_many :akapit_transzy, :order => "transza_id"
60    has_one :akat0, :class_name => "AkapitTranszy",
```

```

61      :conditions => " akapit_transzy_id = (select min(akapit_transzy_id)
        from akapit_transzy atx where " +
62        " atx.akapit_id = akapit_transzy.akapit_id )"

64      has_many :statusy
65      has_one :stat0, :class_name => "Statusy", :conditions =>
66        "statusy_id = (select min( statusy_id ) from statusy sx where
        sx.akapit_id = statusy.akapit_id) "

68      has_many :komentarz, :order => :created_at

70      has_many :token_seg, :class_name => "Token", :order => :kolejnosc,
71      :conditions => "sg_choice_id is not null"
72      # Uwaga! To wybiera także tokeny odrzuconych wariantów segmentacji,
        i tak właśnie ma i musi być.

74      has_many :token_sup, :class_name => "Token", :order => :kolejnosc,
75      :conditions => "superancja is not null"

77      has_many :sg_choice

79      has_many :chosen_interps, :class_name => "Token",
80      :order => "kolejnosc", :conditions => {:czy_interp => true, :chosen
        => true}

82      belongs_to :path

84      has_many :końce_zdań, :class_name => "Token", :order => :kolejnosc,
85      :conditions => { :czy_konczy_zdanie => true, :chosen => true }

87      has_many :token_sens, :class_name => "Token", :order => :kolejnosc,
88      :conditions => " exists ( select interpretacja_id from interpretacja
        i " +
89        " where i.disamb='t' and i.cz_m_leksem_id is not null " +
90        " and i.token_id=token.token_id )"

94      def treść_z_tokenów( tokenstwo = :token )
95        t166=self.send( tokenstwo ).reload
96        (t166.collect{|t167|
97          t167.orth +
98          ( if t167.czy_konczy_zdanie? : ' '
99            else "
100              end ) +
101          (if t167.ns_nastepuje?
102            # metoda bez znaku zapytania może źle działać przy niektórych
            bazach danych:
103            # tam, gdzie fałsz jest przechowywany jako o ( if o => true )
104            "
105            else ' '
106            end)
107          }).join.strip + if tokenstwo == :token
108            "
109            else '...'
110            end
111      end # of treść_z_tokenów

```

```

114 def treść_pełna_spacjowana
115   t168=self.token_all.reload
116   (t168.collect{|tok169| tok169.orth }).join(' ').strip
117 end # of tresc

120 def treść_idowana
121   t170=self.token.reload
122   t170.collect {|tok171| "#{tok171.orth}({tok171.id})" }.join
123 end # of treść_idowana

126 def tokslex
127   # zwraca tablicę tokenów z interpretacjami:
128   ## [
129   ##   [orth1, interp11, interp12, ...]
130   ##   [orth2, interp21, interp22, ...]
131   ## ...]
132   self.token.reload.collect{|token|
133     t1=Array.new
134     t1«(token.orth)
135     token.interpretacja.each{|interpretacja|
136       t1«(if interpretacja.disamb
137           "&#x25b8; "
138           # >, x10fb georgian par. separator, x2023 triangle right, x203b X z czterema
139           kropkami (reference mark)
140           # x25b8 small triangle right (bigger than the previous)
141
142           else
143             "&nbsp; &nbsp;"
144           end +
145           if interpretacja.disamb
146             "<b>"
147           else
148             ""
149           end +
150             "<i>" +interpretacja.leksem.lemat + "</i> " +
151             interpretacja.leksem.klasa_gram.klasa_gram_ozn +
152             interpretacja.reszta_tagu +
153             if interpretacja.disamb
154               "</b>"
155             else
156               ""
157             end
158           end
159         )
160       }
161     t1
162   }
163 end # of tokslex.

168 def self.zaanotowane( status_symbol=:zakończony, clauses_hash={} )
169   # wynik ma dodatkową kolumnę uzid równą mniejszemu z identyf.
170   użytkownika (o ile takowe występują)

```



```

170 # Porządkujemy wg corpus_path_id, żeby wypisywać w
    chunkach-ścieżkach
171 # pierwszy argument może też być dwuelementową tablicą [:status_od,
    :status_do]
172 # a drugi argument to hash pseudo-opcji :pogrupuj, :where i :having jak
    na razie.
173 # Uwaga. Ta metoda zwróci także akapit odrzucone. Aby pominąć
    odrzucone, należy
174 # podać pseudo-opcję where => " zt.odrzucone = 'f'"
175
176 logger.info "==== szukam zaanotowanych: " + Time.now.to_s( :db )
177
178 ssym = status_symbol
179 ch = clauses_hash
180
181 if status_symbol.kind_of?( Hash )
182   ssym = ( status_symbol[ :status_symbol ] || :zakończony )
183   ch = status_symbol
184
185 elsif PoziomyAnotacji.poziomy_x.include?( ssym )
186   ch[ :poziom ], ssym = ssym, :zakończony
187 end
188
189 betw = AkapitTranszy.status_betw( ssym )
190
191 if poz = ( clauses_hash[ :poziom ] || clauses_hash[ :poziomy ] )
192   # jeśli hasz parametrów zawiera klucz :poziom lub :poziomy,
    wybierzemy tylko akapity o zadanym statusie na tym poziomie/tych
    poziomach anotacji, a jeśli nie zawiera – to o zadanym statusie na
    jakimkolwiek poziomie anotacji.
193   kriterion = [ poz ].flatten.collect { |poz| " ( s.#{poz} #{betw}
    ) " }.join(" or ")
194 else
195   kriterion = PoziomyAnotacji.działające.collect{ |poz|
196     " ( s.#{poz} #{betw} ) " }.
197   join(" or ")
198 end
199
200 ##      uzidy = PoziomyAnotacji.poziomy_x.collect {      }
201 # co Autor miał na myśli? chyba nic szczególnego, skoro z tego
    zrezygnował.
202
203
204 if true # spróbowałem prewybierać tokeny, ale to wydłużyło ogólny
    czas renderowania widoku (i czas wykonania tego zapytania oczywiście).
205   sql = "select akapit.* " +
206     " from akapit inner join statusy s using( akapit_id ) " +
207     " where ( #{kriterion} ) " +
208     if clauses_hash[:where] : " and " + clauses_hash[:where]
209     else "" end +
210     " group by akapit_id " +
211     if clauses_hash[:having] : " having " + clauses_hash[:having]
212     else "" end +
213     " order by akapit_id "
214 else

```

```

215     h195 = {
216         ##           :select => " akapit.*, token.* ",
217         # nie ma potrzeby: Rails wybierają wszystkie kolumny z akapit
218         # i token, nazywając je zresztą po swojemu

219         :from => " (akapit inner join statusy s using( akapit_id )) ",
220         ## +
221         ##           " inner join token using( akapit_id ) ",
222         # token zostanie dołączony przez :nclude

223         :conditions =>          " ( #{kriterion} ) " + ## and
224         token.chosen='t'
225         # warunek na token wybrany znajdzie się w warunku złączenia
226         # dodanego przez :include
227         if clauses_hash[:where] : " and " + clauses_hash[:where]
228         else "" end ,

229         :group => " akapit.akapit_id, token_id " + # bez token_id
230         # zwracało po jednym przypadkowym tokenie
231         if clauses_hash[:having] : " having " + clauses_hash[:having]
232         else "" end ,

233         ##           :order => " akapit.akapit_id, token.kolejnosc",
234         :include => :token
235     }
236 end

237 logger.info h195.inspect
238 ##     wynik195 = self.find( :all, h195 )
239 wynik195 = self.find_by_sql( sql )
240 logger.info "### po wyszukaniu zaanotowanych: " + Time.now.to_s(
241 :db ) +
242     " (znalazłam #{wynik195.size})"
243 return wynik195

244 end # of self.zaanotowane.

245 def self.quasi_transza( limits )
246     self.zaanotowane(
247         :zakończony,
248         :where => " akapit.akapit_id between #{limits[0]}
249             and #{limits[1]} " )
250 end # of quasi_transza.

251 def xml_id
252     if self.original_id : return self.original_id
253     else return self.akapit_id.to_s
254     end
255 end

256 def self.init_minmax
257     if (not defined?( @@min_id )) or (not defined?( @@max_id ))
258         @@min_id = self.find_by_sql("select min(akapit_id) as lim from
259             akapit")[0].lim.to_i

```

```

271      @@max_id = self.find_by_sql("select max(akapit_id) as lim from
      akapit")[0].lim.to_i
272    end
273  end

275  def self.min_id
276    init_minmax
277    return @@min_id
278  end

280  def self.max_id
281    init_minmax
282    return @@max_id
283  end

286  def dolicz_sensy_zweryf( param=nil )
287    mozna=true
288    unless param
289      zt0=akapit_transzy[0]
290      if zt.ma_status?(:>=, :zweryfikowany) and
291        not zt.odrzucone?
292        param = zt0.akat_hash
293      else
294        mozna=false
295      end
296    end# of unless param.

298    if mozna
299      token.each{ |tok172|
300        sa = SensAnot.znajdz( tok172.token_id, param )

302        if sa
303          sasl = sa.sensy_leksemu
304          sasl.sens_zweryf_count +=1
305          sasl.save
306        end # of if sa,
307      }
308    end# of if mozna,
309  end# of dolicz_sensy_zweryf.

312  def self.autozatwierdź_segmentation ## ( akapit_od, akapit_do )
313    # procedura, która akapit o jednoznacznej segmentacji oznaczy (nada
314    status) jako zweryfikowany na poziomie segmentacji
315    if self.find( :first ) and self.
316      find_by_sql(
317        [ "select count( distinct akapit_id ) as ile from
          statusy where " +
318          " not exists (select * from sg_choice sgc where " +
319          " sgc.akapit_id=statusy.akapit_id) and " +
320          " segmentation>=?",
321          AkapitTranszy::STATI[ :zweryfikowany ] ]
322      )[ 0 ].ile.to_i <
323    self.

```

```

324         find_by_sql(
325             "select count( distinct akapit_id ) as ile from
              statusy where " +
326             " not exists (select * from sg_choice sgc where " +
327             " sgc.akapit_id=statusy.akapit_id) " ) [0].ile.to_i
328 # jeżeli liczba akapitów sg-zweryfikowanych jest mniejsza od (liczby
akapitów zatranszowanych pomniejszonych o liczbę akapitów nie
mających wariantów segmentacyjnych) to:

330     natror_id = Uzytkownik.natror.id

332     self.find( :all, :conditions =>
333         " (select min(segmentation) from statusy s " +
334         " where s.akapit_id=akapit.akapit_id)" +
335         "<16 and not exists " +
336         " (select * from sg_choice sgc where
          sgc.akapit_id=akapit.akapit_id) "
337         ## { :akapit_id => akapit_od..akapit_do }
338     ).each { |aka|
339         if aka.akat0 and not aka.token_segm.reload[ 0 ]
340             puts "#{aka.id}:"
341             aka.akapit_transzy.each{ |akat|
342                 if akat.ma_status?( :segmentation, :<, :zweryfikowany )
343                     akat.set_status( :segmentation, :zweryfikowany, :tylko_to,
344                                     natror_id )
345                     akat.propaguj_dopuszczalność( :segmentation )
346                     puts "      #{akat.id}"
347                 end
348             }
349             else puts "#{aka.id} ma warianty segmentacyjne lub nie jest
          zatranszowany"
350             end
351         }# of each aka,
352         return true
353     else
354         return false
355     end# of main if
356 end# of self.autozatwierdź_segmentation.

359 def self.ile_do_segmentacji
360     # działa dobrze: sprawdzone przez select Count(* ) from (select distinct
akapit_id from token where czy_segm='t');
361     self.find_all.collect do |aka|
362         if aka.token_segm.reload[ 0 ]
363             true
364         else nil
365         end
366     end. # of collect aka
367     compact.size
368 end# of self.ile_do_segmentacji.

371 ## dopuść_posegmentowane

```

```

373      #
375      def self.zaznacz_końce_ostatnich_zdań ## ( akapit_od, akapit_do )
377          logger.info "==== zaznaczam końce_ostatnich_zdań"
378          końce_nie_zaznaczone = " select akapit_id " +
379              " from akapit left outer join koniec_zdania_annot using(
380                  akapit_id ) " +
381              " where koniec_zdania_annot_id is null " ## +
382          ## " and akapit_id between #{akapit_od} and #{akapit_do}"
384          if self.find_by_sql( " select 1 where exists
              (#{końce_nie_zaznaczone})" )[0]

386              natror_id = Uzytkownik.natror.id
387              AkapitTranszy.find( :first ) # iżby zainicjować

389              logger.info "==== naprawę zaznaczam"
390              self.connection.execute " insert into koniec_zdania_annot " +
391                  " ( created_at, akapit_id, akapit_transzy_id, uzytkownik_id,
392                      token_id ) " +
393                  " select current_timestamp, akapit_id, atr.akapit_transzy_id,
394                      #{natror_id}, token_id " +
395                  " from akapit_transzy atr inner join token t using( akapit_id )
396                  " +
397                  " where token_id in " +
398                  " ( select max( token_id ) from token " +
399                  " where akapit_id in (#{końce_nie_zaznaczone}) group by
400                      akapit_id) "
401              # tu jest śliskość: opieramy się na porządku id-ów, a tymczasem
402              # decydująca jest kolejność. Ale robimy to wyłącznie na danych
403              # dziewiczych, dla których kolejność i token_id są zgodne.
404              true
405          end # of main if
406      end # of self.zaznacz_końce_ostatnich_zdań

407      def self.zdezambiguuj_interpunkcje ## ( akapit_od, akapit_do )
408          punteggiature_ambigui = "select interpretacja_id " +
409              " from interpretacja inner join token using( token_id ) " +
410              " where czy_interp='t' and ( disamb is null or disamb='f') " ##
411          +
412          ## " and akapit_id between #{akapit_od} and #{akapit_do}"

413          if self.find_by_sql( " select 1 as esistenza where exists
              (#{punteggiature_ambigui}) " )[0]
414              self.connection.execute " update interpretacja set disamb='t' " +
415                  " where interpretacja_id in ( #{punteggiature_ambigui} ) "
416              true
417          end
418      end # of self.zdezambiguuj_interpunkcje.

419      ## zdezambiguuj_interpunkcje

420      text = ' @tokXXXs = nil def tokXXXs unless @tokXXXs and
421          @tokXXXs.size == self.token.reload.size @tokXXXs =

```

```

    self.token.collect{ |tNRNR| tNRNR.XXX } end @tokXXXs end'
    # of tokids i tokorths instance methods.

425 eval text.gsub( 'XXX', 'id' ).gsub( 'NRNR', '173' )
426 eval text.gsub( 'XXX', 'orth' ).gsub( 'NRNR', '174' )

428 def czy_posegm?
429     self.akat0.ma_status?( :segmentation, :>=, :zweryfikowany )
430 end

434 def self.autozatwierdź_word_senses
435     # procedura, która akapit o jednoznacznej segmentacji oznaczy (nada
        status) jako zweryfikowany na poziomie segmentacji
436     Statusy.autozatwierdź_word_senses
437 end# of self.autozatwierdź_word_senses.

440 def tree3levels
441     # Zwrócimy drzewo, tj. tablicę tablic: tablicę zdań, z których każde jest
        tablicą tokenów i sg_choice'ów

443     tok_choice = self.token.dup

445     self.token_segm.each { |ts|
446         tok_choice[ self.tokids.index( ts.id ) ] = ts.sg_choice if
            ts.chosen?
447     } # of each token_segm
448     # teraz tok_choice jest tablicą tokenów i choice'ów.

450     prev_index = 0
451     sentences = self.końce_zdań.collect { |tok176|
452         # zastępujemy token kończący zdanie przez tablicę tokenów od
            poprzedniego końca zdania do tego.
453         curr_index = self.tokids.index( tok176.id )
454         # uwaga: metoda token zwraca tablicę złożoną tylko z tokenów
            wybranych, ale możemy być pewni, że każda niejednoznaczność
            segmentacyjna (element choice) ma w tej tablicy reprezentanta.

456         ## puts "### akapit #{self.id}, tok176.id==#{tok176.id}"

459         difference = curr_index - prev_index + 1
460         start_index, prev_index = prev_index, curr_index + 1
461         tok_choice[ start_index, difference ].uniq # uniq żeby usunąć
            ewentualne duplikaty sg_choice'ów.
462     } # of końce_zdań.collect

464 end # of tree3levels

467 def tree2levels
468     # Zwrócimy drzewo, tj. tablicę tablic: tablicę zdań, z których każde jest
        tablicą (wybranych!) tokenów.

470     prev_index = 0

472     tok = self.token

474     sentences = self.końce_zdań.collect { |tok176|

```

```

475     # zastępujemy token kończący zdanie przez tablicę tokenów od
    poprzedniego końca zdania do tego.
476     curr_index = self.tokids.index( tok176.id )
477     # uwaga: metoda token zwraca tablicę złożoną tylko z tokenów
    wybranych, ale możemy być pewni, że każda niejednoznaczność
    segmentacyjna (element choice) ma w tej tablicy reprezentanta.

479     ##          puts "### akapit #{self.id}, tok176.id==#{tok176.id}"

482     difference = curr_index - prev_index + 1
483     start_index, prev_index = prev_index, curr_index + 1
484     tok[ start_index, difference ]
485 } # of końce_zdań.collect

487 end # of tree2levels

492 def responsables3levels
493     ## [segmentation, sentences, morphosyntactic ]
    [ :segmentation, :sentences, :morphosyntactic ].collect{ |poz|
495         self.akapit_transzy.collect{ |a|
496             if anorka = a.anotatorka( poz )
497                 anorka.login
498             end
499         }.compact.join( ' + ' )
500     }
501 }
502 end # of responsables3levels

505 def tree_word_senses
506     # Zwrócimy drzewo, tj. tablicę tablic: tablicę zdań sensowanych,
    z których każde jest tablicą tokenów sensownych.
507     # każde zdanie-tabliczka tokenów sensownych jest ubogacone na końcu
    tokenem swego końca (który nie musi być sensowny, a gdyby był
    sensowny, to zostaje powtórzony).
508     tok_sens = self.token_sens.dup

510     if tok_sens[0]
511         # aby podzielić tokeny sensowne na zdania, patrzymy na ich kolejność
    i odcinamy te, u których jest ona mniejsza niż kolejność kolejnego
    końca zdania.
512         sentences = self.końce_zdań.collect { |tok176|
513             # zastępujemy token kończący zdanie przez tablicę tokenów
    sensownych od poprzedniego końca zdania do tego.
514             curr_sent = []

516             while tok_sens[0] and tok_sens[0].kolejność <= tok176.kolejność
517                 curr_sent << tok_sens.delete_at( 0 )
518             end # of while
519             if curr_sent[0]
520                 curr_sent << tok176 # to zwraca curr_sent.
521             else
522                 nil
523             end
524         }.compact # of końce_zdań.collect

```

```

526         else
527             return nil
528         end
529     end # of tree_word_senses

532     def responsables_word_senses
533         # zwraca string loginów anotatorek WSD „Kizia + Mizia”.
534         self.akapit_transzy.collect{ |a|
535             if anorka = a.anotorka( :word_senses )
536                 anorka.login
537             end
538         }.compact.join( ' + ' )
539     end # of responsables_word_senses

542     def self.skomentowane( wszystkie=nil )

544         self.find_by_sql( "select j0.*, min(k2.akapit_transzy_id) as
                    akapit_transzy_id from (" +
545                             " select akapit.*, max(k.created_at) as kcrea " +
546                             " from akapit inner join komentarz k
                    using(akapit_id) " +
547                             " unless wszystkie : " where k.nowy = 't' " else "
                    end +
548                             " group by akapit_id) as j0 " +
549                             " inner join komentarz k2 " +
550                             " on j0.kcrea=k2.created_at and
                    j0.akapit_id=k2.akapit_id " +
551                             " group by j0.akapit_id, kcrea order by kcrea "
                    )
552     end # of self.skomentowane

556     @@strictease = Proc.new { |orths|
557         if orths[-1] == :strict
558             strict = orths.delete_at( -1 )
559         else
560             strict = nil
561         end
562         strict
563     }

565     @@orths_multijoin = Proc.new { |o215|
566         sq215 = ""
567         # sprawdziłem doświadczalnie: zwraca jedną kolumnę imieniem akapit_id.
568         o215.each_index { |i215|
569             if i215 > 0
570                 sq215 += " inner join "
571             end
572             sq215 += " token t#{i215} "
573             if i215 > 0
574                 sq215 += " on t#{i215-1}.akapit_id = t#{i215}.akapit_id "
575             end
576         }

```



```

577     sq215 += " where "
578     o215.each_index { |i215|
579       if i215 > 0
580         sq215 += " and "
581       end
582       sq215 += " t#{i215}.orth=? "
583     }
584     sq215
585   }

588   @@orths_intersect = Proc.new { |o215|
589     sq215 = ""
590     # sprawdziłem doświadczalnie: zwraca jedną kolumnę imieniem akapit_id.
591     o215.each_index { |i215|
592       if i215 > 0
593         sq215 += " intersect "
594       end
595       sq215 += " select akapit_id from token where orth=? "
596     }
597     sq215
598   }

601   def self.znajdź_po_orthach( * orths )
602     strict = @@strictease.call( orths )
603     o = orths.join(' ').split
604     ##      sq = " select t0.akapit_id from " + @@orths_multijoin.call(
605       o )
606     sq = @@orths_intersect.call( o )
607     a = Akapit.find_by_sql( [ " select * from akapit where akapit_id in
608       (#{sq})", o ].flatten )
609     if strict
610       a.collect { |x| if Regexp.new( Regexp.escape( o.join(' ') ) ) =~
611         x.treść_pełna_spacjowana : x end }.compact
612     else
613       a
614     end
615   end # of znajdź_po_orthach

616   def są_orthy?( * orths )
617     strict = @@strictease.call( orths )
618     o = orths.join(' ').split
619     ##      sq = " select 1 from " + @@orths_multijoin.call( o ) + "
620       and t0.akapit_id=? "
621     sq = " select 1 from ( " + @@orths_intersect.call( o ) + " ) where
622       akapit_id=? "
623     tid = Token.find_by_sql( [ sq, o, self.id ].flatten )[0]

624     unless tid
625       return false
626     else # czyli zapytanie zwróciło rekord
627       if strict

```

```

628         if Regexp.new( Regexp.escape( o .join(' ') ) ) =~
        self.treść_pełna_spacjowana
629             return true
630         else return false
631         end
632     else # nie ściśle, tj. nie musi być dopasowane w tej kolejności
633         return true
634     end
635 end
636 end # of sprawdź_orthy

640 def superancje
641     # zwracamy hasz indeksowany symbolami poziomów, którego
        wartości są tablice tokenów zasuperowanych na danym poziomie.
642     supy177 = Hash.new

644     self.token( :refresh ).each{ |tok177|
645         if ts177 = tok177.superancja
646             ts177.each { |s177|             (supy177[s177] ||= []) « tok177 }
647         end
648     }

650     supy177

652 end # of superancje

655 @atrids, @trids = nil, nil

657 def init_atrids
658     unless @atrids and @trids
659         @atrids, @trids = [], []
660         self.akapit_transzy.each { |atr|
661             @atrids « atr.id
662             @trids « atr.transza_id
663         }
664         @atrids_join = @atrids.join(', ')
665         @trids_join = @trids.join(', ')
666         true
667     end
668 end

671 def atrids_join
672     init_atrids
673     @atrids_join
674 end

676 def trids_join
677     init_atrids
678     @trids_join
679 end

681 # dopuść_wsd jest metodą klasy Statusy, odpalaną przy jej inicjacji.

683 PF_OFFSET = 10

```

```

684      # możliwe, że dla różnych poziomów anotacji ten offset powinien być
        rozmaity.

686      def set_primafalsa(      ,      )
687          tids = self.tokids
688          if      .kind_of?( Token )
689              =      .id
690          end
691          primafalsa_index = tids.index(      ).to_i
692          if ( prima_off = primafalsa_index - PF_OFFSET ) >= 0
693              [ :primafalsa ] = tids[ prima_off ]
694          end
695      end # of set_primafalsa

698      def primus_sensibilis
699          # uwaga! to działa wyłącznie po weryfikacji na poziomie MS.
700          ##      Token.find_by_sql( ["select * from token where
        kolejnosc=(select min( kolejnosc ) from token t inner join
        interpretacja i using( token_id) where i.disamb='t' and
        i.cz_m_leksem_id is not null and t.akapit_id=?)", self.id ] )[0]
702          self.token_sens.reload[0]
703      end # of primus_sensibilis

706      def self.ile_pobranych
707          self.find_by_sql(
708              " select count( distinct akapit_id ) as ile from
        akapit_transzy a " +
709              " inner join transza t using( transza_id ) " +
710              " where t.uzytkownik_id is not null "
711          )[0].ile.to_i
712      end # of self.ile_pobranych

715      def update_treści
716          self.tresc = self.treść_z_tokenów
717          logger.info "#### treść: " + self.tresc
718          self.incipit = self.treść_z_tokenów( :token_incipny )
719          logger.info "#### incipit: " + self.incipit
720          self.save!
721      end

724      def self.update_treści_null
725          ret203 = false
726          self.find_all_by_incipit( nil ).each { |aka203|
727              puts "incipczę #{aka203.id}"
728              aka203.update_treści
729              ret203 = true unless ret203
730          }
731          return ret203
732      end

735      # self.usuń_puste jest niepotrzebne: wypływka się generuje i dla pustych.
736      # w przypadku decyzji o użyciu tych metod do czasu obsługi płatnej
        doliczyć 15 min.

```

```

738     def self.puste
739         where213 = " where akapit_id in (select akapit_id from akapit a
              where not exists " +
740             " (select * from token where akapit_id=a.akapit_id))"
741         self.find_by_sql( " select * from akapit " + where213 )
742     end # of self.puste

744     def self.usuń_puste
745         c213 = self.connection
746         where213 = " where akapit_id in (select akapit_id from akapit a
              where not exists " +
747             " (select * from token where akapit_id=a.akapit_id))"

749         c213.execute( "delete from statusy " + where213 )
750         c213.execute( "delete from akapit_transzy " + where213 )
751         c213.execute( "delete from akapit " + where213 )

753     end # of self.usuń_puste

756     end # of class.

759     ## # Local Variables:
760     ## # mode: ruby
761     ## # End:

```

### **akapit\_transzy.rb**

```

1      # == Schema Information
2      # Schema version: 51
3      #
4      # Table name: akapit_transzy
5      #
6      # akapit_transzy_id :integer primary key
7      # akapit_id :integer not null
8      # transza_id :integer not null
9      # status :integer default(0), not null
10     # odrzucony :boolean
11     # uzytkownik_id :integer
12     # created_at :timestamp
13     # updated_at :timestamp
14     # blizniaczy_id :integer
15     #

39     require 'ramkowanie'

42     class AkapitTranszy < ActiveRecord::Base

44         belongs_to :akapit
45         belongs_to :transza
46         ## has_many :komentarz, :order => :komentarz_id
48         # nie has many, tylko subtelniej, bo będziemy wybierać komentarze do
         akapit akapit,
49         # a nie do akapit_transzy, albo do uzytkownik-a (może się zdarzyć,
50         # w przypadku odebrania transzy, że komentarze do akapit_transzy nie
         są danego

```

```

51      # użytkownika. Wtedy na dzień dobry nie chcemy ich pokazywać.
52      # Z kolei dopóki akapit nie jest zweryfikowany, akapit_transzy nie ma
      przypisanego uzytkownik-a.
53      has_many :fraza_anot, :order => "token_id_pocz, token_id_kon",
      :dependent => :destroy
54      # podwójne porządkowanie fraz ze względu na przyszłe dopuszczenie
      zagnieżdżeń
55      # klauzula :dependent => destroy jest konieczna, by działała metoda
      delete_all.

58      ## has_one :finitywna, :class_name => "FrazaAnot",
60      ## :conditions => {:fraza_typ_id => FrazaTyp.find_id( 'VP' )}

63      has_one :bliźniaczy, :class_name => "AkapitTranszy",
64      :foreign_key => "blizniaczy_id"

67      has_many :interpretacja_anot, :order => "token_id", :dependent =>
      :destroy
68      has_many :sens_anot, :order => "token_id", :dependent => :destroy

71      has_one :statusy

73      has_many :sg_variant_anot,
74      :order => "sg_variant_id", :dependent => :destroy

76      has_many :koniec_zdania_anot, :order => "token_id", :dependent =>
      :destroy

78      def sg_choice
79          self.akapit.sg_choice
80      end

83      def ma_Status?( comparison, status_symb)
84          # używana w application.rb
85          return self.status.send( comparison, STATI[status_symb] )
86      end

90      def self.status1_2( status_symbol )
91          ## logger.info "#### AKapitTranszy.status1_2 wołany od
      #{status_symbol.inspect}"

94          # :anotaśne i :oczekujące obsługujemy w status_betw.

96          if status_symbol == :zakończony
97              status1 = STATI[ :zweryfikowany ]
98              status2 = STATI[ :zweryfikowany ] + 1000

100          elsif status_symbol == :rozbieżny
101              status1 = STATI[ :zatwierdzony ] + 1
102              status2 = STATI[ :zweryfikowany ] - 1

104          elsif status_symbol == :podsądny
105              status1 = STATI[ :do_osądzania ]
106              status2 = STATI[ :osądzany ]

108          elsif status_symbol.kind_of?(Array)
109              status1 = STATI[ status_symbol[0] ]

```

```

110         status2 = STATI[ status_symbol[1] ]
112     else
113         status1 = STATI[ status_symbol ]
114         status2 = status1
115     end
116     wynik = [status1, status2]
117     ##      logger.info "                               #{wynik.inspect}"
119     wynik
120 end # of self.status1_2

123 def self.status_est?( le_status_en_question, comparison,
le_symbol_de_status )
124     le_status_en_question.send( comparison, STATI[le_symbol_de_status]
)
125 end

128 def self.editable_par_annotateur?( lestatus )
129     # tę metodę wołamy by sprawdzić, czy status akapitu pozwala
anotatorowi go anotaować. Zarządca/Arbiter zawsze może go anotać, o
ile jest na danym poziomie dopuszczony do anotacji.
130     self.status_est?( lestatus, :<, :do_osądzania ) and
131     self.status_est?( lestatus, :>=, :dopuszczony )
132 end

134 def self.confirmant?(lestatus)
135     # tak ma być, bo przy wyświetlaniu listy akapitów pytamy, czy ma być
"oglądaj" czy "anotuj"
136     # więc :zweryfikowany też ma tu wpadać.
137     self.status_est?( lestatus, :>=, :zatwierdzony )
138 end

141 if DlaEli.nk

143     @@poziomy_x = PoziomyAnotacji.poziomy_x
145     @@poziomy_działające = PoziomyAnotacji.działające

147     STATI = PoziomyAnotacji::STATI
148     STATI_obniz = PoziomyAnotacji::STATI_obniz
149     STATI_podnies = PoziomyAnotacji::STATI_podnies
150     STATI_tekst = PoziomyAnotacji::STATI_tekst
151     STATI_kr = PoziomyAnotacji::STATI_kr

154     def statusek( poziom )
155         self.statusy.send( poziom )
156     end # of statusek

158     def ma_status?( poziom, comparison, status_symb )
159         if status_symb == :anotaśne
160             return [ STATI[:dopuszczony], STATI[ :do_poprawki ] ].include?(
self.statusek( poziom ) )
161         else
162             return self.statusek( poziom ).send( comparison,
STATI[status_symb] )

```

```

163     end
164 end

166 def status_html
167     akats = self.statusy
168     stekst = @@poziomy_działające.collect{ |poz|
169         PoziomyAnotacji::NAZWY[ poz ][:kr] + ':<b>' +
170         STATI_kr[akats.send( poz )] +
171         if akats.send( "#{poz}_odrzucony?" )
172             " ODRZUCONY!"
173         else "
174         end + '</b>'
175     }.join(' | ')
176 end

179 def edytowalny_przez_anotatora?( poziom )
180     self.ma_status?( poziom, :>=, :dopuszczony ) and
181     self.ma_status?( poziom, :<, :do_osądenia )
182 end# of edytowalny_przez_anotatora?.

185 def self.poziomy_dostępne( akat_id, rola_id, parhasz={} )
186     self.find( akat_id ).poziomy_dostępne( rola_id, parhasz )
187 end# of self.poziomy_dostępne

190 def poziomy_dostępne( rola_id, parhasz={} )
191     if Rola.anotator?( rola_id )
192         pozdost = []
193         @@poziomy_działające.each{ |poz|
194             pozdost « poz if self.edytowalny_przez_anotatora?( poz ) }
195     elsif Rola.zarządca?( rola_id )
196         pozdost = self.poziomy_o_statusie( :>=, :dopuszczony )
197         pozdost &= self.poziomy_o_statusie( :<, :zweryfikowany ) if
            parhasz[ :jak_anotator ]
198         pozdost -= [ :segmentation ] if # segmentacji raz przyklepanej
            nie można zmienić.
199         parhasz[ :segmentation_specjalnie ] and
200         self.ma_status?( :segmentation, :>=, :zweryfikowany )
201     else pozdost = []
202 end

204     pozdost &= PoziomyAnotacji.działające
205     if phpp = parhasz[ :poziomy_pokażne ]
206         pozdost &= phpp
207 end

209     return pozdost
210 end# of poziomy_dostępne

213 def poziomy_anotaśne( rola_id, parhasz={} )
214     if Rola.anotator?( rola_id )
215         pozano = []
216         @@poziomy_działające.each{ |poz|
217             if self.edytowalny_przez_anotatora?( poz ) and
                self.ma_status?( poz, :whatever, :anotaśne )

```

```

218         pozano « poz
219     end
220 }
221 elsif Rola.zarzadca?( rola_id )
222     pozano = self.poziomy_o_statusie( :>=, :dopuszczony )
223     pozano &= self.poziomy_o_statusie( :<, :zweryfikowany ) if
224     parhasz[ :jak_annotator ]
225     pozano -= [ :segmentation ] if # segmentacji raz przyklepanej nie
226     można zmienić.
227     parhasz[ :segmentation_specjalnie ] and
228     self.ma_status?( :segmentation, :>=, :zweryfikowany )
229 else pozano = []
230 end
231
232 pozano &= PoziomyAnotacji.działające
233 if phpp = parhasz[ :poziomy_pokażne ]
234     pozano &= phpp
235 end
236
237 return pozano
238 end# of poziomy_dostępne
239
240 def poziomy_max_annot
241
242 end # of poziomy_max_annot
243
244 def poziomy_o_statusie( comparison_symbol, status_symbol )
245     @@poziomy_x.collect{ |poz|
246         poz if self.ma_status?( poz, comparison_symbol, status_symbol
247         )
248     }.compact
249 end # of poziomy_dopuszczone
250
251 end# of if nk
252
253 # informacja, czy akapit zostało odrzucony, jest w kolumnie odrzucony
254
255 def get_uzid( session_uzid=nil )
256     # zakładamy, że użytkownik akapit transzy nie zmieni się w trakcie
257     trwania tej instancji
258     unless @uzid
259         @uzid = self.uzytownik_id
260         @uzid ||= self.transza.uzytownik_id
261         unless @uzid
262             ## logger.info "### #AkapitTranszy.get_uzid : @uzid is
263             nil for @akat with id #{self.id}, assuming session[:uzid]
264             (==#{session_uzid.inspect})"
265             @uzid = session_uzid
266         end
267     end
268 end
269
270 return @uzid
271 end
272
273 def editable_par?( uzytkownik )

```



```

271     if uzytkownik.kind_of?( Uzytkownik )
272         u = uzytkownik
273     else u = Uzytkownik.find( uzytkownik )
274     end

276     if u.id == self.get_uzid : return true
277     else
278         if u.rola_id == Rola.rid( :zarzadca ) : return true
279         else return false
280     end
281 end
282 end # of editable_par?.

285 def akat_hash
286     unless @akat_hash
287         @akat_hash = { :akapit_transzy_id => self.id }.freeze
288     end
289     return @akat_hash
290 end# of akat_hash.

293 def niezdezamb?
294     # jeśli wszystko zdezambiguowane, zwrócimy nil.
295     # jeśli nie wszystko zdezamb, zwrócimy komunikat o tym, które tokeny
    są niejednoznaczne
296     niejed = ""
297     self.akapit.token.collect { |tok153|
298         dissize = tok153.disamb( self.akat_hash ).size
299         if dissize != 1 : niejed.z_przecinkiem!(
300             "token
                &#0132;#{tok153.orth}&#0148;
                niezdezambiguowany" )

301         @jednakze_mozna_zatwierdzic = false
302     }# of token.collect |tok153|
303     if niejed == "" : return nil
304     else return niejed
305     end
306 end

309 def niesensowny?
310     nil
311 end # of niesensowny nie dla Eli

315 def niefrazowny?(zdziw_sie = nil)
316     niefraz = ""
317     param = self.akat_hash
318     # DlaEli wszystkie tokeny nie-interp mają być we frazach
319     # i ma być co najmniej jedna fraza werbalna.
320     # dla AP nie mogą być wszystkie we frazach, bo nie ma frazy finitywnej.
321     # Ale zdziwimy się, jeśli poza frazami więcej niż jeden.
322     # 2008/02/29 wyłączamy zdziwienie u AP, gdy któryś token poza frazą.
323     if DlaEli.x
324         nie_we_frazie = 0

```

```

325     self.akapit.token.each{ |tok165|
326       fr = self.fraza(tok165.token_id)
327       kg = tok165.disamb( self.akat_hash
328         )[0].leksem.klasa_gram.klasa_gram_ozn
329       if kg != "interp" and kg != "qub" and not fr
330         nie_we_frazie += 1
331       if zdziw_sie
332         niefraz.z_przecinkiem!(
333           "token &#0132;#{tok165.orth}&#0148; jest
334             poza frazą")
335         end
336       end
337     }
338   end
339
340   # unless DlaEli.x
341   #   niefraz = "" if (not zdziw_sie) or nie_we_frazie <= 1
342   # end
343
344   # i każda fraza ma oznaczone co trzeba
345   self.fraza_anot.each {|fraa|
346     fraa_noti = (
347       if fraa.token_id_pocz == fraa.token_id_kon
348         "faza &#0132;#{fraa.token_pocz.orth}&#0148; "
349       else
350         "faza &#0132;#{fraa.token_pocz.orth} " +
351         if fraa.token_id_kon - fraa.token_id_pocz > 1 :
352           "&#0151; " else "" end +
353           "#{fraa.token_kon.orth}&#0148; "
354       end )
355     fraa_notice = fraa_noti + "nie ma "
356     if (not fraa.token_id_pocz) or (not fraa.token_id_kon) or
357       (not fraa.fraza_typ_id)
358       niefraz.z_przecinkiem!(fraa_notice + "typu")
359       @jednakze_mozna_zatwierdzic = false
360     else
361       ft = fraa.fraza_typ
362       if DlaEli.x and ft
363         if ( ft.typ_symbol == "VP" and not
364           (0..1).include?(fraa.vp_typ))
365           niefraz.z_przecinkiem!( fraa_notice + "neg/pos")
366           @jednakze_mozna_zatwierdzic = false
367         end
368         if ( ft.typ_symbol == "sie" and not
369           FrazaAnot.sie_zakres.include?(fraa.sie_typ))
370           niefraz.z_przecinkiem!( fraa_notice + "typu
371             &#0132;się&#0148;")
372           @jednakze_mozna_zatwierdzic = false
373         end
374       end
375     end # of if DlaEli

```

```

370     if (not fraa.token_id_synhead)
371       if ft.moze_bez_glowy?( :synh )
372         niefraz.z_przecinkiem!(fraa_notice + "synh") if zdziw_sie
373       else
374         niefraz.z_przecinkiem!(fraa_notice + "synh")
375         @jednakze_mozna_zatwierdzic = false
376       end
377     else # ma głowę nietypową lub niedopuszczalną
378       synh = fraa.token_synhead
379       dopg = ft.dopuszcza_glowe?( :synh, synh, param)
380       if dopg
381         if zdziw_sie
382           niefraz.z_przecinkiem!(fraa_noti +
383             "ma nietypową synh:
              &#0132;#{synh.orth}&#0148;") if
              dopg[0] == 0
384         end
385       else # nie dopuszcza takiej głowy
386         niefraz.z_przecinkiem!(fraa_noti +
387           "ma niedopuszczalną synh:
              &#0132;#{synh.orth}&#0148;")
388         @jednakze_mozna_zatwierdzic = false
389       end # of if dopg
390     end # of ma / nie ma synh

392     if (not fraa.token_id_semhead)
393       if ft.moze_bez_glowy?( :semh )
394         niefraz.z_przecinkiem!(fraa_notice + "semh") if zdziw_sie
395       else
396         niefraz.z_przecinkiem!(fraa_notice + "semh")
397         @jednakze_mozna_zatwierdzic = false
398       end
399     else # ma semh nietypową lub niedopuszczalną
400       semh = fraa.token_semhead
401       dopg = ft.dopuszcza_glowe?( :semh, semh, param)
402       if dopg
403         if zdziw_sie
404           niefraz.z_przecinkiem!(fraa_noti +
405             "ma nietypową semh:
              &#0132;#{semh.orth}&#0148;") if
              dopg[0] == 0
406         end
407       else # nie dopuszcza takiej głowy
408         niefraz.z_przecinkiem!(fraa_noti +
409           "ma niedopuszczalną semh:
              &#0132;#{semh.orth}&#0148;")
410         @jednakze_mozna_zatwierdzic = false
411       end # of if dopg
412     end # of nie ma semh
413     fraa_notice.chomp!("nie ma ")

```

```

414      # == "frazza „...—.” ”
416      # raczej ma byc jednoglowa
417      if fraa.token_id_synhead != fraa.token_id_semhead
418        if ft.jednoglowa?
419          niefraz.z_przecinkiem!( fraa_notice + "ma synh &#8800;
            semh" )
420          @jednakze_mozna_zatwierdzic = false
421        elsif ft.raczej_jednoglowa? and zdziw_sie
422          # nie musi jednoglowa, ale ma ten sam pref. typ semh co synh
423          niefraz.z_przecinkiem!( fraa_notice + "ma synh &#8800;
            semh" )
424        end
425      end # end of raczej jednoglowa
426    end # of czy nie ma typu i granic (else)
427  } # of each fraa

429  if DlaEli.x # dla Eli ma miec fraze czasownikowa (VP)
430    ma_VP = ( self.fraza_anot.collect { |fraa|
431      if fraa.fraza_typ and fraa.fraza_typ.typ_symbol == "VP"
432        1; else nil; end }.compact.size > 0)
433    unless ma_VP
434      niefraz.z_przecinkiem!("brak frazy czasownikowej")
435      @jednakze_mozna_zatwierdzic = false
436    end
437  end

439  if niefraz == "" : return nil
440  else return niefraz
441  end

443  end # of niefrazowny

444  def self.znajdz( zdid, uzid )
445    self.find(
446      :first,
447      :include => :transza,
448      :conditions =>[
449        "akapit_transzy.akapit_id = ? and " +
450        " ( akapit_transzy.uzytownik_id = ? or
451        transza.uzytownik_id = ?) ",
452        zdid, uzid, uzid ]
453      )
454  end # of self.znajdz

455  def pocz_frazy?(tokid)
456    pofra = nil
457    self.fraza_anot(:refresh).each{ |fraa|
458      pofra=fraa if fraa.token_id_pocz == tokid
459    }
460    return pofra
461  end

462  def reszta_frazy?(tokid)

```

```
469     refra=nil
470     self.fraza_annot(:refresh).each{ |fraa|
471       refra = fraa if tokid > fraa.token_id_pocz and tokid <=
         fraa.token_id_kon
472     }
473     return refra
474 end

477 def we_frazie?( tokid )
478   # jeśli token nr tokid nie jest w żadnej frazie tego akat, zwrócimy nil.
479   # jeśli jest, to zwrócimy 0 gdy w środku, 1 na początku, 2 na końcu, 3
     gdy stanowi całą frazę.
480   wefra=nil
481   if 1 == 0
482     self.fraza_annot(:refresh).each{ |fraa|
483       poczid = fraa.token_id_pocz
484       konid = fraa.token_id_kon
485       if (poczid..konid).include?(tokid)
486         if tokid == poczid : wefra = 1
487         else wefra = 0 end
488         if tokid == konid : wefra += 2
489         else wefra +=0 end
490       end
491     }
492     end
493     return wefra
494 end # of we_frazie

496 def indeks_frazy(tokid)
497   ifra= nil
498   self.fraza_annot(:refresh).each{ |fraa|
499     poczid = fraa.token_id_pocz
500     konid = fraa.token_id_kon
501     if (poczid..konid).include?(tokid)
502       ifra = self.fraza_annot.index(fraa)
503     end }
504   return ifra
505 end

509 def fraza(tokid)
510   fra = nil
511   self.fraza_annot.each{ |fraa|
512     fra = fraa if tokid >= fraa.token_id_pocz and tokid <=
         fraa.token_id_kon
513   }
514   return fra
515 end

518 ##   def blizniaczy
520 ##     AkapitTranszy.find( self.blizniaczy_id )
522 ##   end
524 # załatwione przez self-referential has_one.
```

```

527 def weryfikacja( hasz = {} )
528   # wołana przez <anotacja_controller>.zweryfikuj_akapit
529   # zwraca hasz { :werdykt => <y>, :werdykciki => { :<poziom> => <x> },
      :<poziom> => [<id-y>] },
530   # gdzie x == -1 gdy weryfikacja dała na danym poziomie wynik
      niepomyślny,
531   # x == 0 gdy akapit bliźniacze nie zatwierdzony na tym poziomie,
532   # x == 1 gdy wynik na danym poziomie pomyślny,
533   # y = min({x_poziom}_poziomy ∪ {1}).
534   ## dokonca = nil,
535   ##      logger.info "=== hasz do weryfikacji: #{hasz}"
536   ##      logger.info "method #AkapitTranszy.weryfikacja od:
      #{hasz.inspect}"
540   raise "#AkapitTranszy.weryfikacja: :właśnie_zatwierdzamy is nil."
      if hasz[ :właśnie_zatwierdzamy ] == nil
541   protokoluj = hasz[:protokoluj]
542   poziomy = [ hasz[:poziomy] ].flatten.compact

544   rola_id = hasz[:rola_id]
545   rola_id ||= Uzytkownik.find( self.get_uzid(hasz[:session_uzid])
      ).rola_id

547   ##      logger.info "                        poziomy: #{poziomy.inspect}"

550   if poziomy[0]
551     weryf = { :werdykt => 1, :werdykciki => {} }
552   else
553     raise "#AkapitTranszy.weryfikacja: no annotation level given."
554   end

556   właśnie_zatw = hasz[:właśnie_zatwierdzamy]

558   poziomy.each{ |poz|
559     if protokoluj
560       pr = Protokol.zrób_się( self.akapit, poz )
561     else pr = nil
562     end

564     sb = self.bliźniaczy

566     if (
567       ( właśnie_zatw or self.ma_status?( poz, :>=, :zatwierdzony )
        ) and
568       ( sb.ma_status?( poz, :==, :zatwierdzony ) or
569       sb.ma_status?( poz, :==, :po_poprawce ) or
570       Rola.zaradca?( rola_id )
571     ) or
572     ( hasz[ :dokonca ] and
573     ( not właśnie_zatw ) and
574     self.ma_status?( poz, :>=, :zatwierdzony ) and
575     sb.ma_status?( poz, :>=, :zatwierdzony )
576     )

```

```

577      ##      logger.info "@@@ rola_id: #{rola_id}, na poz.
      :#{poz}: mój statusek: #{self.statusek( poz )}, statusek
      bliźniaka #{self.bliźniaczy.statusek( poz )}"
579      ##      logger.info "@@@ weryf_#{poz} #{Time.now}"
581      self.send( "weryf_#{poz}", weryf, pr, protokoluj )
582      ##      logger.info "@@@ weryf_#{poz} #{Time.now}"
584      elsif właśnie_zatw or hasz[ :dokonca ]
585        weryf[ :werdykt ] = 0 if weryf[ :werdykt ] > 0
586        weryf[ :werdykciki ][ poz ] = 0
587      end
588    }# of each poz.

590    primafalsa = poziomy.collect{ |po| [ weryf[ po ] ]
    }.flatten.compact.min
591    if primafalsa # jeśli mamy niezgodność, to umieścimy kotwiczkę trzy
    tokeny wyżej, żeby uwzględnić fixed header.
592      self.akapit.set_primafalsa( weryf, primafalsa)
593    end # of if primafalsa

595    ##      logger.info "@@@@ weryf: " + weryf.inspect
597    imple_superna_gratia( poziomy, weryf ) if właśnie_zatw
598    return weryf

600  end# of weryfikacja.

602  # Tu następują metody sprawdzenia gotowości i weryfikacji akapit na
    poszczególnych poziomach anotacji.

605  def segmentation_niegot?( rola_id, zdziw_sie ) # zdziwienie nad
    sentences zależy od roli
606    wynik =[]
607    ##      wynik_orth = []
609    self.sg_choice.each { |sgc|
610      chosen_variant_met = false
611      sgc.sg_variant.each{ |sgv|
612        if sgvan = SgVariantAnot.find( :first, :conditions =>
613          { :sg_variant_id => sgv.id,
614            :akapit_transzy_id => self.id } )
615          if chosen_variant_met
616            if sgvan.chosen?
617              sgv.token.each{ |tok154| wynik « [ tok154.id, tok154.orth
618                ]
619              }
620            end
621          else chosen_variant_met = sgvan.chosen?
622          end
623        else
624          sgv.token.each{ |tok155| wynik « [ tok155.id, tok155.orth ] }
625        end
626      }# of each variant
627    unless chosen_variant_met
      sgc.token.each{ |tok156| wynik « [ tok156.id, tok156.orth ] }
    end
  end

```

```

628         end
629     }# of each choice

631     wynik.uniq!
632     wynik, wynik_orth = wynik.dx, wynik.dy

634     if wynik[0]
635         return "Nie mogę zatwierdzić akapitu, w którym token(y)
        #{wynik_orth.join(', ')} ma(ją) nie ujednoznaczoną
        segmentację."
636     else return "
637     end
638 end # of segmentation_niegot?.

641 def wyniknij( poziom, weryf, wynik, werdykcik )
642     wynik = nil unless wynik[0]
643     weryf[:werdykt] = werdykcik if weryf[:werdykt] > werdykcik
644     weryf[:werdykciki][poziom] = werdykcik
645     weryf[poziom] = wynik if wynik
646 end # of wyniknij.

649 def weryf_segmentation( weryf, pr, protokoluj )
650     ##          logger.info "### weryf_segmentation"
651     wynik = []
652     werdykcik = 1

655     self.sg_choice.each { |sgc|
656         sgc.sg_variant.each{ |sgv|
657             currverd = nil
658             sgvars = sgv.sg_variant_anot
659             if ( sgvars.size == 2 ) and ( sgvars[ 0 ].chosen? != sgvars[ 1
                ].chosen? )
660                 sgv.token.each{ |tok157| wynik « tok157.id }
661                 currverd = -1

663             elsif sgvars.size != 2 # weryfikujemy, a więc akapit został
                sprawdzony, że po naszej stronie ma wszystko zaanotowane.
                Zakładam, że jak nie 2, to 1.
664                 currverd = -1 # jeżeli akapit bliźniaczy ma status c.n.
                :zatwierdzony, a ta segmentacja nie ma pary, to coś jest nie halo.
665                 sgv.token.each{ |tok158| wynik « tok158.id }
666             end

668             if currverd == -1 and protokoluj
669                 pr.protokoluj_segmentation( self, sgc, sgv )
670                 pr.protokoluj_segmentation( self.bliźniaczy, sgc, sgv )
671             end

673             werdykcik = currverd if werdykcik > -1 and currverd == -1
674         }# of each variant
675     }# of each choice

677     self.wyniknij( :segmentation, weryf, wynik, werdykcik )
679 end# of weryf_segmentation.

```



```

682  def ujednoznaczniej_segmentacje
683    tokens = self.akapit.token
684    tokids = tokens.collect {|tok159| tok159.id}
685    self.segmentacja_anot.each{ |sa|
686      # przejrzymy tokeny, przestawimy im czy_segmem na false,
687      # a te, które należy połączyć, najpierw zaprotokołujemy, jakie były,
        a potem połączymy.
688      i = tokids.index( sa.token_id )
689      token_nast = tokens[ i+1 ]
690    }
691  end # of ujednoznaczniej_segmentacje

694  def zdziwko( zdziw_sie, komunikat )
695    if komunikat == " or komunikat == nil
696      return "
697    else
698      if zdziw_sie and @jednakze_mozna_zatwierdzic
699        return "Dziwię się, że " + komunikat + "."
700      else
701        return "Nie mogę zatwierdzić akapitu, w którym " + komunikat
          + "!"
702      end
703    end
704  end # of zdziwko.

707  def sentences_niegot?( rola_id, zdziw_sie=nil )
708    # zdziwimy się, jeśli znak interpunkcyjny, który zwykle kończy zdanie,
        nie jest oznaczony jako taki, oraz jeśli ostatni token akapitu nie kończy
        zdania.
709    komunikat = []
710    if zdziw_sie and not Rola.zarzacca?( rola_id )
711      wynik = []
712      wynik_orth = []
713      self.akapit.chosen_interps.each{ |interp|
714        if [ '.', '?', '!' ].include?( interp.orth ) and not
          KoniecZdaniaAnot.znajdz( self, interp )
715          wynik « interp.id
716          wynik_orth « "[#{if ipo = interp.poprzedni : ipo.orth end}]"
            + interp.orth
717        end
718      }
719      if wynik[0]
720        komunikat « "Dziwię się, że token(y) #{wynik_orth.join(' ')}
          nie kończy/ą zdania."
721      end
722    end

724    if not self.koniec_zdania_anot[0]
725      komunikat « "Nie mogę zatwierdzić akapitu, w którym nie
        oznaczono ani jednego końca zdania."
726    end

```

```

728     return komunikat.join(' ')
730 end # of sentences_niegot?

733 def weryf_sentences( weryf, pr, protokoluj )
734   wynik = []
735   werdykcik = 1

737   self.akapit.token.each{ |t160|
738     currverd = nil
739     unless [0,2].include?( t160.koniec_zdania_anot.size )
740       currverd = -1
741       wynik « t160.id
742     end

744     if currverd == -1 and protokoluj
745       pr.protokoluj_sentences( self, t160 )
746       pr.protokoluj_sentences( self.bliźniaczy, t160 )
747     end

749     werdykcik = -1 if currverd == -1 and werdykcik > -1
750   }

752   self.wyniknij( :sentences, weryf, wynik, werdykcik )
753 end

755 def morphosyntactic_niegot?( rola_id, zdziw_sie=nil )
756   self.zdziwko( zdziw_sie, self.niezdezamb? )
757 end # of morphosyntactic_niegot?.

760 def weryf_morphosyntactic( weryf, pr, protokoluj )
761   # sprawdzamy tokeny: dyzambiguacje i sensy
762   wynik = []
763   werdykcik = 1
764   akat = self
765   akath = self.akat_hash
766   bliakat = self.bliźniaczy
767   bliakath = bliakat.akat_hash

769   loggi = {}
770   poprz, nast = Time.now, Time.now
771   logujcz = Proc.new { |w, t|
772     loggi[ w ] ||= 0.0
773     poprz, nast = Time.now, Time.now
774     loggi[w] += nast - poprz
775     ##      logger.info "### {t.id} #{w}: #{nast - poprz} "
776   }

779   self.akapit.token.each { |tok161|
780     currverd = nil
781                                     logujcz.call( :pocz, tok161 )
782     disa = tok161.disambs( akath )[0]
783     blidisa = tok161.disambs( bliakath )[0]
784     # dyzambiguacje są jednoznaczne, skoro akapit zatwierdzony+.

```

```

785                                     logujcz.call( :po_znajdź_disa,
                                           tok161 )
786     unless disa and blidisa
787         wynik « tok161.id
788         currverd = -1
789         # weryfikujemy, a więc jeśli w bliźniaczym brakuje którejś
           dezambiguacji, to znaczy, że nie jest dokończony. A jednak ma
           statusek c.n. zatwierdzony, czyli coś nie halo.
790                                     logujcz.call( :nie_ma_disa )
791     else # są disa i blidisa
792         # diclass « disa.class
793         # diclass « blidisa.class
794
795         if (disa.leksem_id != blidisa.leksem_id) or (disa.reszta_tagu
           != blidisa.reszta_tagu)
796             currverd = -1
797             wynik « tok161.id
798         end
799     end
800     end
801
802     if currverd == -1 and protokoluj
803         pr.protokoluj_morphosyntactic( akat, tok161, disa )
804         pr.protokoluj_morphosyntactic( bliakat, tok161, blidisa )
805     end
806
807     werdykcik = -1 if currverd == -1 and werdykcik > -1
808
809 } # of each token.
810
811 logger.info "=== weryf_morphosyntactic #{loggi.inspect}"
812 ## logger.info " (bli)disa.class: #{diclass.uniq.inspect}"
813 logger.info "=== weryf_morphosyntactic przed wyniknij:
           #{Time.now}"
814
815 wyniknij( :morphosyntactic, weryf, wynik, werdykcik )
816
817 end# of weryf_morphosyntactic.
818
819 def word_senses_niegot?( rola_id, zdziw_sie ) # zdziwienie nad
sentences zależy od roli
820
821     wynik =[]
822     ##     wynik_orth = []
823     wszystkie_tokeny_osensowane = true
824     self.akapit.token.each { |tok188|
825         if tok188.do_sensu?( self.akat_hash ) and not tok188.the_sens(
           self.akat_hash )
826             wynik « [ tok188.id, tok188.orth ]
827         end}
828
829
830     wynik, wynik_orth = wynik.dx, wynik.dy
831
832     if wynik[0]
833         return "Nie mogę zatwierdzić akapitu, w którym nie wskazano sensu
           tokenu(ów) #{wynik_orth.join(', ')}."
834     else return "

```

```

835     end
836 end # of word_senses_niegot?.

839 def weryf_word_senses( weryf, pr, protokoluj )
840   ##      logger.info "=== weryf_segmentation"
842   wynik = []
843   werdykcik = 1

845   self.akapit.token.each {|tok189|
846     currverd = nil
847     if tok189.do_sensu?( self.akat_hash ) or tok189.do_sensu?(
848       self.bliźniaczy.akat_hash )
849       # Korzystamy z tego, że WSD są następnikiem MS, wobec czego obie
850       # interpretacje morfosyntaktyczne każdego tokenu są zgodne i nie
851       # musimy tego sprawdzać.
852       if ( the_sens189a = tok189.the_sens( self.akat_hash )) !=
853         ( the_sens189b = tok189.the_sens( self.bliźniaczy.akat_hash
854           ))
855         # zresztą, gdyby nie były zgodne, to ich the_sensy będą się różnić.
856         wynik « tok189.id
857         currverd = -1
858       end
859     end
860   end

861   if currverd == -1 and protokoluj
862     pr.protokoluj_word_senses( self, tok189, the_sens189a )
863     pr.protokoluj_segmentation( self.bliźniaczy, tok189,
864       the_sens189b )
865   end

866   werdykcik = currverd   if werdykcik > -1 and currverd == -1
867 }

868   self.wyniknij( :word_senses, weryf, wynik, werdykcik )

869 end# of weryf_segmentation.

871 def self.status_betw( status )
872   # dopuszczalne wartości: symbole statusów: :nie_dopuszczony,
873   # :dopuszczony itd. oraz :zakończony (l.p.) :anotaśne, oczekujące (l.mn.),
874   # bo chodzi o mnogą liczbę statusów.
875   if status == :whatever
876     betw = " not null "
877   elsif status == :anotaśne
878     betw = " in (#{STATI[:anotaśne].join(', ')}) "
879   elsif status == :oczekujące
880     betw = " in (#{STATI[:oczekujące].join(', ')}) "
881   else
882     status1, status2 = status1_2( status )
883     if (status2 - status1) % 1000 == 0
884       betw = " > #{status1 - 1} "
885     else
886       betw = " between #{status1} and #{status2} "

```

```

885         end
886     end
887     betw
888 end # of self.status_betw

891 def self.ile_zaanotowanych( id=nil, status=:zakończony,
format=:inspect )

893     betw = self.status_betw( status )

895     if 1==1 # selecty sprawdzone jako działające
896         szkatulki = @@poziomy_działające.collect{ |poz|
897             " sum( case when s.#{poz} " + betw +
898             " then 1 else 0 end ) as #{poz}"}.join(", ")

900         if id

902             if id.kind_of?( Integer ) or ( uzid = id[ :uzid ] )
903                 uzid ||= id
904                 transza_clause, transza_id = ", nil
905                 esquelle =
906                 Proc.new { |poz| [ "select count(*) as ile from statusy s
                    inner join " +
907                     " transza t using( transza_id ) " +
908                     " where #{poz}" + betw +
909                     " and (s.#{poz}_uzid = ? or ( s.#{poz}_uzid
                    is null and t.uzytownik_id=?)) ",
                    uzid, uzid ] }

911             else
912                 transza_id = id[:transza_id]
913                 esquelle =
914                 Proc.new { |poz| [ "select count(*) as ile from statusy " +
                    " where #{poz}" + betw +
915                     ' and transza_id in (?) ', transza_id ] }

917             end
918             ilezaa = @@poziomy_działające.collect{ |poz|
919                 self.find_by_sql( esquelle.call( poz ) )[0].ile.to_i
920             }
921         else # not uzid
922             rekord = self.find_by_sql( "select " + szkatulki + " from
                    statusy s " )[0]
923             ilezaa = @@poziomy_działające.collect{ |poz|
924                 rekord.send( poz ).to_i/2
925             }
926         end

928     else # not 1==1 – alternatywne selecty, najprawdopodobniej równie
dobrze.

930         if uzid
931             ilezaa = @@poziomy_działające.collect{ |poz|
932                 self.find_by_sql([ "select count(*) as ile from
                    akapit_transzy akat " +

```

```

933         " inner join statusy s on
          akat.akapit_transzy_id=s.akapit_transzy_id "
          +
934         " where s.#{poz} #{betw} and
          akat.uzytownik_id = ? ", uzid ])[0].ile.to_i

935     }
936     else
937         ilezaa = @@poziomy_działające.collect{ |poz|
938             self.find_by_sql( "select count(*) as ile from statusy s " +
939                             " where s.#{poz} #{betw} " ))[0].ile.to_i/2
940         }
941     end
942 end
943 return ilezaa.send( format )
944 end# of self.ile_zaanotowanych.

947 def self.ile_zaanotowanych_total( uzid=nil, status=:zakończony )
948     betw = self.status_betw( status )

950     unless uzid
951         kriterion = @@poziomy_działające.collect{ |poz|
952             " ( s.#{poz} #{betw} ) " }.
953             join(" or ")

955         sql = "select count(*) as ile from statusy s where " + kriterion

957     else # when uzid is given
958         kriterion = @@poziomy_działające.collect{ |poz|
959             " ( s.#{poz} #{betw} and ( s.#{poz}_uzid=? or t.uzytownik_id=?
          )) " }.
960             join(" or ")
961         sql = [ "select count(*) as ile from statusy s inner join " +
962                 " transza t using( transza_id ) where " + kriterion ]
963         ( @@poziomy_działające.size * 2 ).times{ |i|
964             sql « uzid
965         }
966     end # of if uzid or not

968     ##      logger.info "#### " + sql.inspect

971     ilezaa = self.find_by_sql( sql )[0].ile.to_i

973     ilezaa /= 2     if status == :podeśdny

975     ilezaa.to_s
976     # to ma być string

978 end # of self.ile_zaanotowanych_total

983 def statuski
984     self.statusy.statuski
985 end # of statuski

988 def set_status( poziom, status_symbol, tylko_to=false, uzid=nil )
989     # dołączamy czwarty, opcjonalny parametr uzid, żeby można było zadać
    id użytkownika przy automatycznym ujednoznacznieniu segmentacji.

```

```

990      # Na koniec patrzymy, czy przypadkiem nie zmienia się „duży status”, tj.
      pole status w rekordzie akapit_transzy – które ma być równe
      najmniejszemu statuskowi.
991      # tu nie powinno być transakcji: jest wołane już w jakiejś transakcji.
992      logger.info "=== set_status \"akat\"=>\"#{self.id}\": poziom
      #{poziom.inspect}, ssym #{status_symbol.inspect} tylko_to
      #{tylko_to.inspect} uzid #{uzid.inspect}"
993      akat = self
994      bliakat = akat.bliźniaczy
995      akats=self.statusy
996      bliakats = bliakat.statusy
997      akats.set_poziom( poziom, status_symbol, :bez_uS ) # w tym jest
      propaguj_dopuszczalność.
998      bliakats.set_poziom( poziom, status_symbol, :bez_uS ) unless
      tylko_to
999      if status_symbol == :zweryfikowany or
1000        ( (not akats.uzid( poziom )) and status_symbol == :osądzony )
1001        # test sprawdza tylko akat, a nie bliakat, ale dzięki transakcji
1002        # możemy zakładać, że not akat.uzytownik_id wtw not
        bliakat.uzytownik_id
1003        if uzid : bluzid = uzid
1004        else bluzid=nil
1005        end
1006        uzid ||= akat.transza.uzytownik_id
1007        akats.set_uzid( poziom , uzid )
1008        unless tylko_to
1009          bluzid ||= bliakat.transza.uzytownik_id
1010          bliakats.reload.set_uzid( poziom, bluzid )
1011        end
1012      end
1013      akats.save!
1014      bliakats.save! unless tylko_to
1015
1016      akat.update_Status( akats )
1017      bliakat.update_Status( bliakats ) unless tylko_to
1018    end # of set_status.
1019
1020    def update_Status( akats=nil )
1021      # Chcemy, by niezmiennikiem bazy danych było, że w kolumnie status
      tabeli akapit_transzy znajduje się najmniejszy ze statusków danego
      akapitu transzy.
1022      ## logger.info "==== update_Status pre: akat #{self.id}:
      #{self.statuski.inspect} -> #{self.status}"
1023
1024      # 2009/7/8 mój umysł otworzył się i osiągnąłem oświecenie:
      zmartwiłem się bowiem, że ustalanie Statusu ze statusków trzeba
      uzależnić od konkretnych poziomów, czy są one niezależne czy zależne.
      Otóż nie trzeba: to dopuszczacz kolejnych poziomów do anotacji dba o to,
      by anotacja na wyższym stała się możliwa dopiero po zaanotowaniu na
      niższym. To zatem, że na jakimś poziomie jest »zatw«, a na innym
      »osądzany«, oznacza, że według dopuszczacza poziomy te są niezależne –
      i update_Status nie musi tego sprawdzać.

```

```

1026     akats ||= self.statusy( :refresh )
1027     akatss = akats.statuski
1028     statuski_annot = akatss & ((0...16).to_a) # przecinamy statuski
        z przedziałem [:dopuszczony, :zweryfikowany]
1029     duzy_Status = statuski_annot.min # było max, 2009/04/10 zmieniłem
        na min zwna nieliniowość porządku poziomów.
1030     dS_popr = akatss & ((10..12).to_a)
1031     duzy_Status = dS_popr.min    if dS_popr[0]

1033     duzy_Status ||= akatss.max # jeśli na wszystkich poziomach jest c.n.
        zweryfikowany, to zwracamy tę wartość.
1034     ##      logger.info "==== self.status: #{self.status.inspect},
        duzy_Status: #{duzy_Status.inspect}"
1036     if self.status != duzy_Status or true # 2009/02/27 okazało się, że
        są różne, a jednak się nie zapisuje.
1037         self.status = duzy_Status
1038         self.save!
1039         ##      logger.info "==== update_Status of akapit transzy #{self.id}"
1041     end
1042     logger.info "==== update_Status post: \"akat\"=>\"#{self.id}\":
        #{self.statusy.reload.statuski.inspect} -> #{self.reload.status},
        bliźniak: #{self.bliźniaczy.statusy.reload.statuski.inspect} ->
        #{self.bliźniaczy.reload.status}"

1044 end # of update_Status.

1047 def self.update_all_Stati( akapit_od=0, akapit_do=10000000000 )
1048     Statusy.dopusć_minimalne
1049     self.find_all.each{ |akat| akat.update_Status
1050         puts "#{akat.id}: #{akat.status}" }
1051     return true
1052 end # of self.update_all_Stati.

1054 def status_co_najmniej=( poziom, status_symbol )
1055     # uwaga! ona nie zapisuje akat.statusy!
1056     akats = self.statusy
1057     akats.set_poziom( poziom,
1058         [[ STATI[status_symbol].to_i,
1059             akats.send( poziom )].max , STATI[:osądzany]].min
        )
1060     # W jednym wypadku zmieniamy także status akapit bliźniaczego: kiedy
        były oba zweryfikowane.
1061     blis = self.bliźniaczy.statusy
1062     if blis.send( poziom ) == STATI[:zweryfikowany]
1063         blis.set_poziom( poziom, STATI[:osądzany] )
1064         blis.save
1065     end
1066 end# of status_co_najmniej=

1069 def propaguj_dopuszczalność( poziom )
1070     self.statusy.propaguj_dopuszczalność( poziom )
1071 end# of propaguj_dopuszczalność

```



```

1074     def komentarz( uzid = nil, czy_tylko_nowe = nil )
1075         if uzid : conds_hash = {
1076             :uzytkownik_id => uzid,
1077             :akapit_id => self.akapit_id }
1078
1079         else conds_hash = { :akapit_id => self.akapit_id }
1080         end # of if uzid
1081
1082         conds_hash.update( { :nowy => true } ) if czy_tylko_nowe
1083
1084         Komentarz.find(:all, :conditions => conds_hash, :order =>
            :created_at )
1085     end # of komentarz
1086
1087     def skopiuj_na_bliźniaczy( session_uzid, * poziomy )
1088         # nie obudowujemy tego transakcją tutaj, bo tę metodę będziemy wołali
1089         # w transakcji
1090         # w <anotacja_controller>.zatwierdz_akapit
1091
1092         akat = self
1093         @bliakat = akat.bliźniaczy
1094         @uzid = akat.get_uzid( session_uzid )
1095         @bluzid = @bliakat.get_uzid( session_uzid )
1096         @akath = akat.akat_hash
1097         @bliakath = @bliakat.akat_hash
1098
1099         @bliakat.odrzucony = akat.odrzucony
1100         # przy osądzeniu akapit przypiszemy bliźniaczemu jego prawnego
1101         # anotatora,
1102         # ale to w momencie ustalania statusu.
1103
1104         poziomy.each{ |poz| self.send( "skopiuj_#{poz}" )
1105             akat.statusy.set_uzid( poz, @uzid )
1106             @bliakat.statusy.set_uzid( poz, @bluzid )
1107         }
1108
1109         # zapisz bliźniacze
1110         @bliakat.save!
1111         logger.info "### skopiuj na bliźniaczy wykonano, *poziomy:
1112             #{poziomy.inspect}"
1113     end # of skopiuj_na_bliźniaczy.
1114
1115     def skopiuj_segmentation
1116         @bliakat.sg_variant_anot.delete_all
1117         self.sg_variant_anot.each{ |sgva|
1118             @bliakat.sg_variant_anot.create(
1119                 :sg_choice_id => sgva.sg_choice_id,
1120                 :sg_variant_id => sgva.sg_variant_id,
1121                 :chosen => sgva.chosen,
1122                 :akapit_transzy_id => @bliakat,
1123                 :uzytkownik_id => @bluzid )
1124         }
1125     end # of skopiuj_segmentation.
1126
1127     def skopiuj_sentences

```

```

1128 @bliakat.koniec_zdania_annot.delete_all
1129 self.koniec_zdania_annot.each{ |kz|
1130   @bliakat.koniec_zdania_annot.create(
1131     :akapit_id => kz.akapit_id,
1132     :akapit_transzy_id => @bliakat,
1133     :uzytkownik_id => @bluzid,
1134     :token_id => kz.token_id
1135   )
1136 }
1137 end# of skopiuuj_sentences

1140 def skopiuuj_morphosyntactic
1141   @bliakat.interpretacja_annot.delete_all
1142   self.interpretacja_annot.each{ |ia|
1143     @bliakat.interpretacja_annot.create(
1144       :uzytkownik_id => @bluzid,
1145       :token_id => ia.token_id,
1146       :interpretacja_id =>
1147         ia.interpretacja_id
1148     )
1149   }
1150 end # of skopiuuj_morphosyntactic.

1152 # Metody wykonujące dodatkowe operacje gdy zdanie zostało
zweryfikowane na odp. poziomie.

1154 def commit_segmentation
1155   # oznaczamy akapit jako ujednoznaczniiony segmentacyjnie (co znaczy,
że został wybrany wariant segmentacji oraz zostały określone granice
zdań).
1156   # zakładam, że wołamy tę metodę z miejsca, w którym już wiemy, że
decyzje segmentacyjne obojga Anotatorów są zgodne. Bierzemy więc ten
(którykolwiek) <akapit_transzy> i przeglądamy jego <sg_variant_annot>

1158   # ujednoznacznienie segmentacji:
1159   # mamy dla każdego tokenu mającego wariant segmentacji mamy
anotowany wariant segmentacji.
1160   ( sa196 = self.akapit ).token_segm.each{ |tok162|
1161     unless tok162.sg_variant_annot( self.akat_hash ).chosen?
1162       tok162.chosen = false
1163       tok162.chosen_updated_at = Time.now
1164       tok162.save!
1165     end
1166     # wszystkie inne tokeny mają .chosen == true .
1167   }# of each token

1169   sa196.update_treści

1171 end# of commit_segmentation.

1174 def commit_sentences
1175   ( sa196 = self.akapit ).token.each{ |tok163|
1176     tok163.czy_konczy_zdanie = (

```

```

1177             if tok163.koniec_zdania_annot[ 0 ] : true
1178             else false
1179             end )
1180         tok163.czy_konczy_zdanie_updated_at = Time.now
1181         tok163.save!
1182     }

1184     sa196.update_treści
1185 end # of commit_sentences.

1188 def commit_morphosyntactic
1189     # w każdym tokenie oznaczamy interpretację wybraną zgodnie przez
1190     # obie anotatorki jako dezambiguację
1191     self.interpretacja_annot.each{ |ia|
1192         iai = ia.interpretacja
1193         iai.disamb = true
1194         iai.save!
1195     }
1196 end# of commit_morphosyntactic.

1197 def commit_word_senses
1198     ##      raise "musimy doliczyć sensy zweryfikowane"
1200     ## self.sens_annot.each { |sa190|
1201     # Nie jestem pewien, czy sens należy twarde notować w tabeli
1202     # interpretacja. Chyba nie.
1203     ##      }
1204 end# of commit_word_senses

1208 def anotorka( poziom )
1209     if uzid = self.statusy.send( "#{poziom}_uzid" )
1210         Uzytkownik.find( uzid )
1211     end
1212 end # of anotorka

1214 def primus_sensibilis
1215     # uwaga! to działa wyłącznie po weryfikacji na poziomie MS.
1216     self.akapit.primus_sensibilis
1217 end

1220 def self.count_diai
1221     self.count_by_sql( "select count( distinct akapit_id ) as ile from
1222     akapit_transzy" )
1223 end

1225 ## def self.unnull_akat_bliźniaki
1227 ## end
1229 # jest w ramkowanie.rb jako metoda klasy ActiveRecord::Base.

1231 def autozatwierdź_word_senses
1232     self.statusy.autozatwierdź_word_senses
1233 end

1236 private

1238 def imple_superna_gratia( poziomy, weryf )

```

```

1239 # Jeżeli stwierdziliśmy rozbieżności, to wryjemy je w granicę kolumny
      superancja tabeli token
1240 # weryf jest weryfikacją akapitu.
1241 # metodę tę wołamy wyłącznie w jednym miejscu: na koniec
      weryfikacja.
1242 # 2009/7/10 a jeśli stwierdzimy brak rozbieżności, to zaklejmy to,
      cośmy wryli dawniej.
1243 tok_su = self.akapit.token_sup
1244 poziomy.each {|poz|
      # zaznaczymy rozbieżności
1245   if weryf[ :werdykciki ][ poz ] == -1
1246     Token.find( weryf[ poz ] ).each { |tok164|
1247       tok164.superancja ||= []
1248       tok164.superancja |= [ poz ]
1249       tok164.save!
1250     } # of each token
1251   end # of if werdykcik -1
1252   # odznaczymy zgodności, ale tylko jeśli oba akaty są po poprawce
1253   if [self.statusy.send( poz ), self.bliźniaczy.statusy.send( poz
1254     )].uniq == [STATI[:po_poprawce]]
1255     logger.info "### odzółcam akat #{self.id} #{Time.now.to_s( :db
      )}"
1256     rozbidy = [weryf[poz]].flatten.compact
1257     tok_su.each { |tsu|
1258       unless rozbidy.include?( tsu.id )
1259         tsu.odzółć( poz )
1260       end
1261     }
1262   end # of if oba po poprawce
1263 }
1264 end # of imple_superna_gratia

1267 # dopuść_wsd jest metodą klasy Statusy, odpalaną przy jej inicjacji.

1270 end # of class.

1272 ## # Local Variables:
1273 ## # mode: ruby
1274 ## # End:

```

### cz\_m.rb

```

1 # == Schema Information
2 # Schema version: 51
3 #
4 # Table name: cz_m
5 #
6 # cz_m_id :integer primary key
7 # cz_m_ozn :text
8 # created_at :timestamp
9 # updated_at :timestamp
10 #

```

```

34  class CzM < ActiveRecord::Base
35    has_many :klasa_gram
36    has_many :sensy
37    has_many :cz_m_leksem

38
39    has_many :fraza_semhead,
40    :class_name => "FrazaTyp" ,
41    :foreign_key => "cz_m_semhead"

42
43    has_many :fraza_synhead,
44    :class_name => "FrazaTyp" ,
45    :foreign_key => "cz_m_synhead"

46
47    # Zapewniam stałe, bo może klient zażyczy sobie polskich nazw w bazie.
48    # To oczywiście są oznaczenia części mowy stosowane w tabeli.
49    # Prawdopodobnie wystarczy NOUN, bo klient życzy sobie anotacji
    semantycznej
50    # tylko dla rzeczowników.
51    NOUN= 'noun'
52    VERB= 'verb'

53
54    @@aliases = {
55      :adj => "adj/adja/adjp/adjc",
56      :noun => "subst/depr",
57      :verb =>
        "pact/ppas/winien/praet/bedzie/fin/impt/aglt/ger/imps/inf/pant/pcon"
58    }

59
60    def self.aliases
61      @@aliases
62    end

63
64    def self.zainicjuj( force=nil )
65      if ( force or not self.find( :first)) and (not CzMLeksem.find(
        :first ))
66        # jeszcze sensy, ale one się wypełniają i przypisują przy sensowlewie
67        self.connection.execute "update klasa_gram set cz_m_id=null"
68        self.connection.execute "update leksem set cz_m_id=null"
69        self.delete_all
70        raise "aborcja starej zawartości tabeli cz_m nie powiodła się"
71        if self.find( :first )
72          @@aliases.each{ |k128, v128|
73            czmid = self.create!( :cz_m_ozn => k128.to_s ).id
74            v128.split( '/' ).each { |kgo|
75              kg = KlasaGram.find_by_klasa_gram_ozn( kgo )
76              kg.cz_m_id = czmid
77              kg.save!
78            }# of each kg
79          }# of each @@aliases pair
80          self.przypisz_leksemom
81          true
82          ## elsif not nie_chodzi_o_czmleksem

```

```

83      ##      raise "CzM.zainicjuj: próba wywołania miesięczki po 12.
          tygodniu (wobec niepustej tabeli cz_m_leksem)!"
85      else
86      false
87      end
88  end # of self.zainicjuj

90  def self.przypisz_leksemom
91      self.connection.execute " update leksem set cz_m_id=" +
92          "(select cz_m_id from klasa_gram k where
          k.klasa_gram_id=leksem.klasa_gram_id)," +
93          " updated_at=current_timestamp, cz_m_przypisana='t'" +
94          " where cz_m_przypisana='f' "
95  end

98  zainicjuj

102 end

cz_m_leksem.rb

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: cz_m_leksem
5  #
6  # cz_m_leksem_id :integer primary key
7  # lemat :text not null
8  # xmlid :text not null
9  # cz_m_id :integer not null
10 # created_at :timestamp
11 # updated_at :timestamp
12 #

36 require 'ramkowanie'

38 class CzMLeksem < ActiveRecord::Base
39     has_many :sensy, :order => :xmlid
40     belongs_to :interpretacja
41     belongs_to :cz_m

44     def self.przypisz_leksemom_i_interpretacjom
45         ##      puts Time.now.to_s( :db )
47         ##      self.pragmas_for_update

50         CzM.przypisz_leksemom

52         join186 = " leksem inner join interpretacja i186 using( leksem_id
            )"

54         self.connection.execute " create temporary view interp_leksem186
            as " +
55             " select interpretacja_id, lemat, cz_m_id from " + join186

57         # puts "temp view created"—chwila
58         namierzacz_czmleksemu = {

```

```

59     Leksem => " where czl.lemat=leksem.lemat and
60     czl.cz_m_id=leksem.cz_m_id ",
61     Interpretacja => " where czl.lemat=( " +
62     " select lemat from interp_leksem186 where
63     interpretacja_id=interpretacja.interpretacja_id) " +
64     " and czl.cz_m_id=( " +
65     " select cz_m_id from interp_leksem186 where
66     interpretacja_id=interpretacja.interpretacja_id) "
67   }
68   # patrzmy na max updated_at rekordu, który ma przypisany nasz id
69   [ Interpretacja, Leksem ].each { |model186|
70     ##      puts model186.to_s
71
72     model186.connection.execute( " update #{model186.table_name} set
73     cz_m_leksem_id = " +
74     "( select cz_m_leksem_id from
75     cz_m_leksem czl " +
76     namierzacz_czmleksemu[ model186 ] +
77     " ), updated_at=current_timestamp,
78     cz_m_leksem_przypisany='t' " +
79     " where cz_m_leksem_przypisany='f' ")
80     # próba zastosowania wbudowanych
81     konwersji (znak zapytania i tablica)
82     spowodowała segmentation fault error
83
84   }# of each model
85
86   self.connection.execute " drop view interp_leksem186 "
87   ##      self.pragmas_normal
88   ##      puts Time.now.to_s( :db )
89   true
90 end# of self.przypisz_się_leksemom_i_interpretacjom
91
92 def self.dopisz_inne
93   bezinne = self.find( :all, :conditions =>
94     "not exists (select * from sensy s where
95     s.cz_m_leksem_id = " +
96     " cz_m_leksem.cz_m_leksem_id and s.xmlid like
97     '%.inne') "
98   )
99   if bezinne[0]
100     ##      pragmas_for_update
101     bezinne.each { |czml199|
102       sen199 = czml199.sensy.create!(
103         :cz_m_id => czml199.cz_m_id,
104         :xmlid => czml199.lemat.odpolszcz
105         + '.INNE',
106         :n => 666666,
107         :short_def => Sensy::INNE_short,
108         :long_def_xml => Sensy::INNE_long,
109         :long_def_html =>
110         Sensy::INNE_long_html

```

```

103                                     )
104     }
106     ##          pragmas_normal
108     return true
109   else
110     return false
111   end
112   ## bezinne[0] != nil
114   end # of self.dopisz_inne
118 end

```

### **dummy1.rb**

```

23 class Dummy1 < ActiveRecord::Base
24   end

```

### **dummy2.rb**

```

23 class Dummy2 < ActiveRecord::Base
24   end

```

### **fraza\_anot.rb**

```

23 # == Schema Information
24 # Schema version: 10
25 #
26 # Table name: fraza_anot
27 #
28 # fraza_anot_id :integer primary key
29 # akapit_transzy_id :integer not null
30 # token_id_pocz :integer
31 # token_id_kon :integer
32 # token_id_synhead :integer
33 # token_id_semhead :integer
34 # fraza_typ_id :integer
35 # vp_typ :integer
36 # sie_typ :integer
37 # created_at :timestamp
38 # updated_at :timestamp
39 #
41 require 'ramkowanie'
44 class FrazaAnot < ActiveRecord::Base
46   raise "zunikalnij zmienne |tok| !"
48   belongs_to :akapit_transzy
49   belongs_to :fraz_a_typ
51   belongs_to :token_pocz,
52     :class_name => "Token" ,
53     :foreign_key => "token_id_pocz"

```



```

55     belongs_to :token_kon,
56     :class_name => "Token" ,
57     :foreign_key => "token_id_kon"

59     belongs_to :token_semhead,
60     :class_name => "Token" ,
61     :foreign_key => "token_id_semhead"

63     belongs_to :token_synhead,
64     :class_name => "Token" ,
65     :foreign_key => "token_id_synhead"

68     TYP_SIE = [
69         ['morfol.', 0],
70         ['zwrotne', 1],
71         ['wzajem.', 2],
72         ['bezosob.', 3]
73     ]

75     include Ramkowanie # definiuje stałe regexp CASES i string UNDEF

78     def self.sie_zakres
79         unless defined?(@@sie_zakres)
80             @@sie_zakres = 0..(FrazzaAnot::TYP_SIE.size - 1)
81         end
82         @@sie_zakres
83     end # of self.sie_zakres

85     def dlugosc
86         self.token_id_kon - self.token_id_pocz + 1
87     end

89     def tokens
90         Token.find(
91             :all,
92             :conditions => {
93                 :token_id => (self.token_id_pocz)..(self.token_id_kon)},
94             :order => :token_id)
95     end

98     def xheads(xhead_sym, param, ftypid=nil)
99         ftypid = self.fraza_typ_id unless ftypid # w warunku, bo wołamy
100         tę metodę w miejscu,
101         # gdy typ frazy dopiero ma zostać przypisany i podajemy go w
102         parametrze.
103         ftyp = FrazzaTyp.find(ftypid)
104         self.tokens.collect{ |tok|
105             ftyp.dopuszcza_glowe?( xhead_sym, tok, param )
106         }.compact.sort
107         # dostaniemy tablicę tablic [ -1/0, token], którą możemy posortować, bo
108         Token ma metodę <=>
109     end # of xheads

110     def synheads(param, ftypid=nil)

```

```

111     self.xheads(:synh, param, ftypid)
112 end

115 def semheads(param, ftypid=nil)
116     self.xheads(:semh, param, ftypid)
117 end

119 if DlaEli.x
120     def finitywna?
121         if self.fraza_typ_id == FrazaTyp.find_id( 'VP' ) : return true
122         else return false
123     end
124     end # of finitywna? dla Eli
125 else
126     def finitywna?
127         return false
128     end # of finitywna? dla AP
129 end

131 def get_uzid
132     # od 2008/09/01 prawdopodobnie niepotrzebna, w każdym razie nie
    używana w tym pliku.
133     self.akapit_transzy.get_uzid
134 end

136 def akat_hash
137     { :akapit_transzy_id => self.akapit_transzy_id }
138 end

141 def reszta_tagu( headsym, param=nil )
142     param ||= self.akat_hash
143     tsh = self.send( headsym )
144     # domyślnie bierzemy głowę semantyczną, jeśli jej nie ma, bierzemy
    syntakt.
145     tsh ||= self.token_synhead
146     if tsh : tsh.disamb( param )[0].reszta_tagu
147     else
148         "
149     end
150 end # of reszta_tagu

152 def przypadek( headsym, param=nil )
153     # symbol głowy zawsze trzeba podać, bo dla prepn będzie to semhead,
    a dla np – synhead
154     param ||= self.akat_hash
155     rt = self.reszta_tagu( headsym, param )
156     rt =~ CASES
157     return $1.to_s
158 end # of przypadek

160 def sem_lemat( param=nil )
161     param ||= self.akat_hash
162     tsh = self.token_semhead

```

```

163     if tsh : tsh.disamb( param )[0].leksem.lemat
164     else "
165     end
166 end # of sem_lemat

169 def syn_lemat( param=nil )
170     param ||= self.akat_hash
171     tsh = self.token_synhead
172     if tsh : tsh.disamb( param )[0].leksem.lemat
173     else "
174     end
175 end # of syn_lemat

177 def tag( full=false, param=nil )
178     # <zabierać :np:nom: :prepn:do:gen: :prepn:w:loc: :sie:>
179     # 0-8 zabierać aff:fin:sg:_:pri:: cpos cpor
180     # [0-3:fw:prepn:w:cisza:loc:: st,
181     # 3-5:sie,
182     # 5-8:fw:prepn:do:czytanie:gen:: czy]
183     # do używania przy „zrzucie na spłaszczoną świgrę”
184     # full idzie tylko w zrzucie na spłaszczoną Świgrę.

186     # ustawiony dla typów fraz EH. Czy będzie potrzebna dla NKJP?

188     param ||= self.akat_hash

190     sft=self.fraza_typ
191     if sft :     typ = sft.typ_symbol.downcase
192     else     typ=nil
193     end

195     if full :     tag0="#{typ}"
196     # na początku było „fw:”, ale Ela go nie chce w zrzucie na spłaszczoną
    Świgrę.
197     else tag0="#{typ}"
198     end

200     semh = self.sem_lemat( param )
201     semh = 'none' if semh == ""

203     case typ

205     when 'adjp' # frazę przymiotnikową uznajemy za realizację wymagania
    rzeczownikowego
206         if full : tag0 = tag0 + ':' + semh + self.reszta_tagu(
            :token_synhead, param )
207         else tag0= 'adjp:' + self.przypadek( :token_synhead, param )
208         end

210     when 'advp'
211         tag0 += ":{semh}"         if full
212         # z jakiegoś powodu we wzorcu są advp bez stopnia przysłówka (bez
            reszty tagu)

214     when 'np'

```

```

215         if full
216             tag0 = tag0 + ':' + semh + self.reszta_tagu( :token_synhead,
                param )
217             tag0 += ':ter' unless tag0 =~ /(pri|sec|ter)$/
218         else tag0 = tag0 + ':' + self.przypadek( :token_synhead, param )
219         end

221     when 'nump' # zakładam, że fraza liczebnikowa realizuje walencję
                rzeczownikową
222         if full : tag0 = tag0 + ':' + self.sem_lemat( param )
                + self.reszta_tagu( :token_synhead, param )
223         # czy w braku głowy semantycznej (rzeczownika) lemat ma zostać
                pusty, czy dać liczebnik?
224         # czy może „NONE”?
225         else tag0='np:' + self.przypadek( :token_synhead, param )
226         end

228     when 'prepadjp'
229         # we frazach prepadjp, prenp i prepnump ignorujemy liczbę i rodzaj,
                dla zgodności z formatem MW/AP, czyli zastępujemy »reszta_tagu«
                przez »:+przypadek«.
230         if full : tag0 = tag0 + ':' +self.syn_lemat( param ) +':'+ semh
                + ':' +
231             self.przypadek( :token_semhead, param )
232         else tag0 = 'prepadjp:' + self.syn_lemat( param )
233             przyp = self.przypadek( :token_semhead, param )
234             tag0 += ':' + przyp if przyp.jlength > 0
235         end

237     when 'prenp'
238         if full : tag0 = tag0 +':'+ self.syn_lemat( param ) +':'+ semh
                + ':' +
239             self.przypadek( :token_semhead, param )
240         else tag0 = tag0 +':'+ self.syn_lemat( param )+':'+
241             self.przypadek( :token_semhead, param )
242         end

244     when 'prepnump' # realizuje walencję prenp
245         if full : tag0 = tag0 +':'+ self.syn_lemat( param ) +':'+ semh
                + ':' +
246             self.przypadek( :token_semhead, param )
247         else tag0 = 'prenp:' + self.syn_lemat( param ) +':'+
248             self.przypadek( :token_semhead, param )
249         end

251     when 'sie' # Ela chce wyróżniać »się« bezosobowe jako frazę luźną.
252         if full
253             if self.sie_typ == 3 : tag0 = 'np:sie:sg:nom:n:ter:: pron'
254                 # »się« bezosobowe zastępuje mianownik.
255                 # sens zwróci ” i nie zostanie dopisany w exsemaival.
256             else tag0 = 'sie'
257             end

```

```

258     else # nie full
259         tag0 = 'np:nom' if self.sie_typ == 3
260     end
261     # w przypadku nie-full całym tagiem »się« nie-bezosobowego jet typ (
    „sie”)

263     when 'vp'
264         if self.vp_typ == 0
265             tag0 = "neg:fin"
266         else tag0 = "aff:fin"
267         end
268         tag0 = tag0 + self.reszta_tagu( :token_synhead, param )

270     else
271         if typ==nil :   tag0=':???'
272         else           raise "nie rozpoznany typ frazy: #{typ}"
273         end
274     end # of case.

276     ##      tag0+= ':' unless full
278     return tag0
279 end # of tag.

282 def sens( param=nil, pole=:sens_opis )
283     # do używania przy „zrzucie na spłaszczoną świgrę”
284     # param dajemy jako arg. opcjonalny, żeby można go było policzyć raz
    na zewnątrz i przekazać.
285     param ||= self.akat_hash
286     s = self.token_semhead
287     s = s.sens( param ) if s
288     if s : return s.sensy.send( pole ) # 2008/07/18 Ela chce pełnym
    słowem
289     # pełnym słowem w Anotatorni, oznaczeniem w zrzucie na spłaszczoną
    Świgrę
290     elsif self.fraza_typ_id == FrazaTyp.find_id( 'sie' ) : return "
291     # co z sensem nump i prepnum, jeśli nie ma rzeczownika? - ma być
    UNDEF
292     # nump —> np itd, ale nie w liście fraz, prawda?
293     else return UNDEF
294     end
295 end# of sens.

298 end # of class

301 ## # Local Variables:
302 ## # mode: ruby
303 ## # End:

```

## frazo\_typ.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: fraza_typ

```

```

5   #
6   # fraza_typ_id :integer primary key
7   # typ_opis :text
8   # typ_symbol :text
9   # synhead_list :text
10  # semhead_list :text
11  # jednoglowa :boolean
12  # created_at :timestamp
13  # updated_at :timestamp
14  #

38  class FrazaTyp < ActiveRecord::Base

40    @@downhash = nil

42    def self.downhash
43      # zwraca hasz ozn => id etykietowany oznaczeniami fraz małymi literami
44      unless @@downhash
45        @@downhash = Hash.new
46        self.find(:all).each{|ft|
47          @@downhash[ft.typ_symbol.downcase] = ft.fraza_typ_id
48        }
49      end
50      @@downhash
51    end # of self.downhash

55    def self.wybor
56      self.find(
57        :all,
58        :order => :typ_symbol).collect { |ft|
59        [ ft.typ_symbol,
60          ft.fraza_typ_id]
61      }
62    end

64    def self.typy_glow
65      self.provide_typy_glow
66      @@typy_glow
67    end

69    def klasy_glowy(head_symbol)
70      if /synh/ =~ head_symbol.to_s
71        head_symbol = :synhead_list
72      elsif /semh/ =~ head_symbol.to_s
73        head_symbol = :semhead_list
74      else raise "nie rozpoznany symbol głowy!"
75      end
76      FrazaTyp.provide_typy_glow
77      kgs = self.send( head_symbol ).split(",").collect{ |ht|
78        @@typy_glow[ht] }
79      # chcemy zwrócić tablicę dwóch tablic: głów typowych i głów
      # nietypowych
80      typowe = kgs[0]

```

```

81     if kgs[1, kgs.size] : nietypowe = kgs[1, kgs.size].flatten.uniq
82     else               nietypowe = []
83     end
84     return [typowe, nietypowe]
85 end # of klasy_glowy

86
87 def dopuszcza_glowe?( head_sym, token, param)
88     # jeśli klasa gram. tokenu jest na liście typowych dla głowy, zwracamy
89     [-1, token, lemat ],
90     # jeśli jest na liście nietypowych, zwracamy [0, token],
91     # jeśli nie ma na żadnej liście, zwracamy nil.
92     if token
93         le = token.disambiguate(param)[0].leksem
94         kg = le.klasa_gram.klasa_gram_ozn
95         lemat_1, lemat0 = le.lemat, le.lemat + " (!)"
96     else
97         kg, lemat_1, lemat0 = nil, "(brak)", "(brak (!))"
98     end
99     kgs = self.klasy_glowy( head_sym )
100    if kgs[0].include?(kg) : return [-1, token, lemat_1]
101    elsif kgs[1].include?(kg) : return [0, token, lemat0]
102    # takie tablice będziemy sortować, więc mamy zdef. metodę <=> egz.
103    # dokładamy lemat, bo będziemy tworzyć listę głów, na której ma być
104    # lemat, a nie orth
105    else return nil
106 end

107
108 def moze_bez_glowy?( head_sym )
109     # bo brak głowy w konkretnych typach fraz AP
110     # nie ma być czymś nietypowym, jeśli dopuszczalny
111     # ale u Eli ma być nietypowy, jeśli na pierwszym miejscu listy
112     x = self.klasy_glowy( head_sym ).flatten.include?( nil )
113     if Eli.x
114         y = self.klasy_glowy( head_sym ).flatten.include?( nil )
115         # tu było coś dziwnego i u Eli nie chciało działać, więc 2008/03/31
116         # dodaję flatten
117     else y = true
118     end
119     if x and y : return -1
120     elsif x and not y : return 0
121     else return nil
122 end

123
124 def bezglowa? # bo konkretny typ fraz może nie mieć obu głów
125     if self.moze_bez_glowy?( :synh) and self.moze_bez_glowy?( :semh)
126         return true
127     else return false
128 end

```

```

129     end
132     def raczej_jednoglowa?
133         self.jednoglowa? || (
134             self.klasy_glowy( :synh )[0] ==
135             self.klasy_glowy( :semh )[0])
136     end
139     private
141     def self.provide_typy_glow
142         unless defined?( @@typy_glow )
143             @@typy_glow = Hash.new
144             klasy_gram = KlasaGram.find(:all).collect{|kg| kg.klasa_gram_ozn
145             }.sort
146             @@typy_glow["any"] = klasy_gram
147             @@typy_glow["none"] = [nil]
148             klasy_gram.each{|kg| @@typy_glow[kg] = [kg] }
149             CzM.find(:all).each{|czm|
150                 @@typy_glow[czm.cz_m_nazwa] = czm.klasa_gram.collect{|kg|
151                     kg.klasa_gram_ozn }.sort
152             }
153         end # of unless defined? @@typy_glow
154     end # of inicjuj_typy_glow.
155
156     @@typy = Hash.new
157     # będzie to hasz { 'VP' => 8, 'AdjP' => <...> } tworzony dynamicznie
158     def self.find_id( typ_symbol )
159         unless @@typy[ typ_symbol]
160             sfind = self.find( :first, :conditions => { :typ_symbol =>
161                 typ_symbol } )
162             if sfind
163                 @@typy[ typ_symbol ] = sfind.fraza_typ_id
164             else puts "brak symbolu ::::#{typ_symbol}::::::"
165             end
166         end
167     end
168     return @@typy[ typ_symbol ]
169     end # of self.find_id
170
171     end # of class
172
173     ## # Local Variables:
174     ## # mode: ruby
175     ## # End:

```

### interpretacja.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: interpretacja
5  #
6  # interpretacja_id :integer primary key
7  # token_id :integer not null

```



```
8   # lex_xmlid :text not null
9   # msd_xmlid :text not null
10  # path_id :integer not null
11  # numer_lex_token :integer
12  # reszta_tagu :text
13  # leksem_id :integer
14  # disamb :boolean
15  # dodana :boolean
16  # cz_m_leksem_id :integer
17  # sensy_id :integer
18  # created_at :timestamp
19  # updated_at :timestamp
20  # cz_m_leksem_przypisany :boolean
21  #

45  require 'ramkowanie'

47  class Interpretacja < ActiveRecord::Base
48    belongs_to :leksem
49    belongs_to :token
50    has_many :interpretacja_anot
51    has_many :sens_anot
52    has_one :sensy

54    belongs_to :cz_m_leksem

56    has_one :msd_opis, :foreign_key => :leksem_id
57    # widok do generowania wypłuwki, migr. 051. (2010/2/25)

59    # nowo tworzonej interpretacji trzeba przypisać cz_m_leksem. Ale to się
    # przy właniu danych odbywa hurtowo. Trzeba więc tylko zadbać o dopisanie
    # go w fc_by_fulltag.

61    def fulltag
62      [self.leksem.klasa_gram.klasa_gram_ozn,
        self.reszta_tagu].compact.join(':')
63    end

66    def self.fc_by_fulltag( tokid, lemat, tag )# find or create by full tag
67      # Uwaga. Ta metoda dotyka także innych klas i być może tworzy ich
        # obiekty.
68      raise "Niepoprawny lemat/tag »#{lemat}«/»#{tag}«" unless
        Tagset.check_tag( tag, lemat )[0]
69      reszta_tagu = tag.split(':')
70      kg = reszta_tagu.delete_at( 0 )
71      reszta_tagu = if reszta_tagu[0] : reszta_tagu.join(':') end
72      # sprawdzamy, czy już jest taka interpretacja:
73      # przede wszystkim – czy jest taki leksem
74      kg_id = KlasaGram.find_by_klasa_gram_ozn( kg ).id
75      conds = {
76        :lemat => lemat,
77        :klasa_gram_id => kg_id,
78        :path_id => Token.find( tokid ).path_id
```

```

79      # tu chodzi o encję leksem, zlokalizowaną do ścieżki korpusu. Ta
      trójka parametrów określa encję leksem jednoznacznie (jest uniczny
      indeks).
80    }
81    le = Leksem.find( :first, :conditions => conds )
82    # Jeśli nie – to go tworzymy:
83    unless le
84      path_id, lex_xmlid, msd_xmlid = self.get_xmlids( tokid, lemat,
      kg_id ) #
85      le = Leksem.create!( conds.update(
86                          :lex_xmlid => lex_xmlid
87                          ) )
88      CzM.przypisz_leksemom
89      le.reload
90    end
91    # A teraz sprawdzamy, czy jest taka interpretacja:
92    conds = {
93      :token_id => tokid,
94      :leksem_id => le.id,
95      :path_id => le.path_id,
96      :reszta_tagu => reszta_tagu
97    }
98    intrpr = self.find( :first, :conditions => conds )
99    unless intrpr
100      unless path_id and lex_xmlid and msd_xmlid # mogły zostać
      pobrane/obliczone w wierszu 649.
101        old_path_id = nil
102        old_path_id = path_id if path_id
103        path_id, lex_xmlid, msd_xmlid = self.get_xmlids( tokid, lemat,
        kg_id )
104        if old_path_id and path_id != old_path_id
105          raise "### #Interpretacja.rb 63: path_id=#{path_id.inspect},
          old_path_id=#{old_path_id.inspect}"
106        end
107      end
108      intrpr = self.create!( conds.dup.update(
109                          :numer_lex_token =>
110                          lex_xmlid.split('.')[0],
111                          :disamb => false, # gdybyśmy tę
112                          :dodana => true,
113                          :lex_xmlid => lex_xmlid,
114                          :msd_xmlid => msd_xmlid,
115                          :path_id => path_id,
116                          :cz_m_leksem_id =>
117                          le.cz_m_leksem_id ,

```

```

116                                     :cz_m_leksem_przypisany => true
117                                     ) )
118     end # of unless intrpr.
119
120     intrpr
121 end # of self.fc_by_fulltag
122
123 def self.zainnij( int, interpretacje )
124     nry_innych_lt = [ 0 ]
125     inne_leksemidy_h = {}
126
127     interpretacje.each { |i2|
128         if i2 != int
129             nry_innych_lt « i2.numer_lex_token
130
131             inne_leksemidy_h[ i2.leksem_id ] = i2.numer_lex_token
132         end
133     }
134     # teraz nr_lt_y zawiera numery lexów naszego tokenu,
135     # zaś lexids_h zawiera te-ż numery etykietowane leksem_id-ami, dla
136     # łatwego wyszukiwania.
137
138     [ nry_innych_lt, inne_leksemidy_h ]
139 end# of self.zainnij
140
141 def zainnij
142     @nry_innych_lt, @inne_leksemidy_h = self.class.zainnij( self,
143         self.token.interpretacja )
144     [ @nry_innych_lt, @inne_leksemidy_h ]
145 end # of instance zainnij
146
147 def self.get_xmlids( tokid, lemat, kgid )
148     # mamy zwrócić tablicę [ path_id, lex_id, msd_id ].
149     # pierwszy – łatwo.
150     # z drugim jest subtelniej: to ma być xmlid elementu <lex>, który ma
151     # być taki sam jak innych interpretacji tego tokenu tej encji leksem, albo
152     # unikalny. I tak się dzieje.
153
154     logger.info "### {tokid.inspect} {lemat.inspect}
155         {kgid.inspect}"
156     path_id = Token.find( tokid ).path_id
157
158     sql1 = " select interpretacja.* " +
159         " from interpretacja inner join leksem using( leksem_id ) " +
160         " where token_id=? "
161     # sql1 zwraca wszystkie interpretacje danego tokenu
162     sql0 = sql1 + " and lemat=? and klasa_gram_id=? " ## and
163     leksem.path_id=?
164     # sql0 zwraca interpretacje tego samego lex-a.
165     ils0 = self.find_by_sql( [sql0, tokid, lemat, kgid ] ) ## ,
166     path_id
167     # IV człon warunku dodany 2009/9/8 po migracji 038, która wedle
168     # mego przekonania powinna była eliminować przypadek innej ścieżki dla
169     # leksemu niż dla tokenu i interpretacji. Ale tego nie robiła, jak boleśnie
170     # okazało się 2009/11/05.

```

```

167 # lex_xmlid ma postać <nr akapitu w pliku>.<nr tokenu w akapicie>.<nr
lex-a w~tokenie>
168 # pierwsze dwa człony numeru są dla danego tokenu takie same.
169 # msd_xmlid jest j.w. plus .<msd_nr>

171 ils1 = []

173 if ils0[0] # znaleźliśmy interpretację(e) tego tokenu o takim leksemie
174   path_id0 = ils0[0].path_id.to_i # to zawsze jest i winna być ta
sama path_id.
175   # skoro znaleźliśmy interpretacje tego samego leksemu, to bierzemy
max id zwiększony o 1.
176   początek_ida = ils0[0].lex_xmlid.split('.')
177   lex_id = początek_ida.delete_at(-1)
178   msd_id = ils0.collect { |il|
179     il.msdxmlid.split('.')[-1].to_i
180   }.compact.max + 1

182   logger.info "=== get_xmlids - gałązka gdy są idy:
msd_id=#{msd_id}"

184 else # nie ma interpretacji o takim leksemie (będziemy mieli nowy
<lex>)
185   logger.info "=== interpretacja w.120 - get_xmlids gałązka gdy
nie ma idów"
186   ils1 = self.find_by_sql( [ sql1, tokid ] )
187   if ils1[0]
188     path_id0 = ils1[0].path_id.to_i # to zawsze jest jedna ścieżka
189     początek_ida = ils1[0].lex_xmlid.split('.')[0,2]

191     nry_lt, h_lt = self.zainnij( nil, ils1 )

193     lex_id = nry_lt.max + 1

195   else # ten token nie ma ani jednej interpretacji
196     ## {<akapit_nr>.<token_nr>.<lex_nr> }
197     ## logger.info "=== interpretacja w.131"
200     początek_ida = [ Token.find( tokid ).akapit_id, tokid ]
201     path_id0 = path_id
202     lex_id = 1
203   end
204   msd_id = 1
205 end

207 unless początek_ida # tzn. gdy ils0 lub ils1 jest niepusta
208   ## logger.info "=== interpretacja w.140"
209   początek_ida = ( ils0 + ils1 ).collect { |il|
210     il.lex_xmlid.split('.') }.max
211   początek_ida[-1] = początek_ida[-1].to_i + 1
212   lex_id = początek_ida.delete_at(-1)
213 end

215 lex_id = (początek_ida + [ lex_id ]).join('.')
216 msd_id = [ lex_id, msd_id ].join('.')

```

```

218     if path_id0 != path_id
219         logger.info "=== niespójność danej path_id? dla tokenu #{tokid}
            i ils0==#{ils0.inspect} i ils1==#{ils1.inspect}:
            path_id0==#{path_id0}, path_id==#{path_id}"
220     ##     logger.info "=== interpretacja w.148 " + [ path_id,
            lex_id, msd_id ].inspect
222     end

224     # 2010/2/18 to powodowało błąd powtórzonych lex_xmlidów w osobnych
    <lex>-ach, a chroniło przed błędem powodowanym przez przypisanie
    leksemu z innej ścieżki.
225     # Ale trzeba sprawdzić, czy znaleziony lex_xmlid jest z naszego lex-a, bo
    jeśli nie, to trzeba utworzyć nowy. Pytanie, czy oprócz nas coś jest w
    „naszym lexie” rozpatrzono wyżej: ils0

227     #
228     ##     if Leksem.find( :first, :conditions => {
229         ##             :path_id => path_id,
230         ##             :lex_xmlid => lex_id } )
231     ##         pocz_lexida = lex_id.split('.')[0,2].join('.')
232     ##         leksid207 = Leksem.find_by_sql( "select lex_xmlid from
            leksem where path_id=#{path_id} and lex_xmlid like
            '#{pocz_lexida}.*' " ).
233     ##         collect { |lex207| lex207.lex_xmlid.split('.')[-1].to_i }.
234     ##         max + 1
235     ##
236     ##         lex_id = "#{pocz_lexida}.#{leksid207}"
237     ##         msd_id = "#{lex_id}.#{msd_id.split('.')[-1]}"
238     ##
239     ##     end # if found leksem of such ids
240
241     return [ path_id, lex_id, msd_id ]
242
243 end # of self.get_xmlids

244
245 def self.odsmiec( conditions = {} )
246     # Usuwanie tagów dodanych a sierocych. Użyte także w klasie Token.
247     conditions.merge!( :dodana => true )
248     smieci = self.find(
249         :all, :conditions => conditions, :include =>
250         :interpretacja_anot
251     ).collect { |inte|
252         inte.id unless inte.interpretacja_anot[0] }.compact
253     self.delete( smieci ) if smieci[0]
254 end# of self.odsmiec

255
256 def cz_m_leksem_id_namierzony
257     self.leksem.cz_m_leksem_id_namierzony
258 end

259 end

260
261 ## # Local Variables:
262 ## # mode: ruby
263 ## # End:

```

**interpretacja\_annot.rb**

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: interpretacja_annot
5  #
6  # interpretacja_annot_id :integer primary key
7  # akapit_transzy_id :integer not null
8  # uzytkownik_id :integer not null
9  # token_id :integer not null
10 # interpretacja_id :integer not null
11 # created_at :timestamp
12 # updated_at :timestamp
13 #

37 class InterpretacjaAnot < ActiveRecord::Base
38   belongs_to :interpretacja
39   belongs_to :uzytkownik
40   belongs_to :token
41   belongs_to :akapit_transzy

44   # usuwanie tagów dodanych a sierocych znajduje się w klasie Token.

46   def bliźniacza
47     @bli_hasz = {
48       :akapit_transzy_id => self.akapit_transzy.blizniaczy_id,
49       :token_id => self.token_id
50     }
51     self.class.find(
52       :first,
53       :conditions => @bli_hasz )
54   end # of bliźniacza

57   def skopiuj_na_bliźniaczę
58     bli = self.bliźniacza # ta met. tworzy @bli_hasz
59     bli ||= self.class.new( @bli_hasz )
60     bli.interpretacja_id = self.interpretacja_id
61     bli.uzytkownik_id = self.uzytkownik_id
62     bli.save!
63   end # of skopiuj_na_bliźniaczę

66 end

69 ## # Local Variables:
70 ## # mode: ruby
71 ## # End:

```

**interpretacja\_export.rb**

```

1  # == Schema Information
2  # Schema version: 51

```

```

3      #
4      # Table name: interpretacja_export
5      #
6      # interpretacja_export_id :integer primary key
7      # interpretacja_id :integer
8      # token_id :integer
9      # lex_xmlid :text
10     # msd_xmlid :text
11     # path_id :integer
12     # numer_lex_token :integer
13     # reszta_tagu :text
14     # leksem_id :integer
15     # disamb :boolean
16     # dodana :boolean
17     # cz_m_leksem_id :integer
18     # sensy_id :integer
19     # created_at :timestamp
20     # updated_at :timestamp
21     # cz_m_leksem_przypisany :boolean
22     # lemat :text
23     # klasa_gram_ozn :text
24     #
50     class InterpretacjaExport < ActiveRecord::Base
51       # klasa oparta na widoku interpretacja_export, pomocniczym do
       # generowania wypływu i tylko do tego: pomija interpretacje dodane, które
       # nie są dezambiguacjami. Dla pełnego widoku jest
       # interpretacja_z_leksemem
56     end # of class

```

### interpretacja\_z\_leksemem.rb

```

1      # == Schema Information
2      # Schema version: 51
3      #
4      # Table name: interpretacja_z_leksemem
5      #
6      # interpretacja_export_id :integer
7      # interpretacja_id :integer
8      # token_id :integer
9      # lex_xmlid :text
10     # msd_xmlid :text
11     # path_id :integer
12     # numer_lex_token :integer
13     # reszta_tagu :text
14     # leksem_id :integer
15     # disamb :boolean
16     # dodana :boolean
17     # cz_m_leksem_id :integer
18     # sensy_id :integer
19     # created_at :timestamp

```

```

20 # updated_at :timestamp
21 # cz_m_leksem_przypisany :boolean
22 # leksem_id:1 :integer
23 # lex_xmlid:1 :text
24 # path_id:1 :integer
25 # lemat :text
26 # klasa_gram_id :integer
27 # cz_m_id :integer
28 # cz_m_leksem_id:1 :integer
29 # created_at:1 :timestamp
30 # updated_at:1 :timestamp
31 # cz_m_przypisana :boolean
32 # cz_m_leksem_przypisany:1 :boolean
33 # klasa_gram_id:1 :integer
34 # cz_m_id:1 :integer
35 # klasa_gram_nazwa :text
36 # klasa_gram_ozn :text
37 # klasa_sem :text
38 # created_at:2 :timestamp
39 # updated_at:2 :timestamp
40 #

66 class InterpretacjaZLeksemem < ActiveRecord::Base
67   # klasa oparta na widoku interpretacja_z_leksemem. Nazwa chyba mówi
   # za siebie. Aha, dołączona jest też klasa gramatyczna.

70 end # of class

```

### **klasa\_gram.rb**

```

1 # == Schema Information
2 # Schema version: 51
3 #
4 # Table name: klasa_gram
5 #
6 # klasa_gram_id :integer primary key
7 # cz_m_id :integer
8 # klasa_gram_nazwa :text
9 # klasa_gram_ozn :text
10 # klasa_sem :text
11 # created_at :timestamp
12 # updated_at :timestamp
13 #

37 class KlasaGram < ActiveRecord::Base
38   belongs_to :cz_m

40   def self.zsynchronizuj_z_tagsetem
41     Tagset.find( :first )
42     esquelle = "select lewe from tagset t left join klasa_gram k on
   t.lewe=k.klasa_gram_ozn where t.typ='pos' and k.klasa_gram_ozn is
   null "
43     if self.find_by_sql( esquelle )[0]

```



```

44     self.connection.execute " insert into klasa_gram( created_at,
      klasa_gram_ozn ) " + esquelle.sub( "select", "select
      current_timestamp, " )
45     end
46
47 end # of self.zsynchronizuj_z_tagsetem
48
49 zsynchronizuj_z_tagsetem
50
51 INTERP = 'interp'
52
53 @@nulla = nil
54
55 def self.nulla
56     unless @@nulla
57         unless @@nulla = self.find( :first, :conditions =>
58             "klasa_gram_ozn is null" )
59             @@nulla = self.create!( {
60                 :cz_m_id => nil,
61                 :klasa_gram_nazwa => nil,
62                 :klasa_gram_ozn => nil,
63                 :klasa_sem => nil })
64             end # of the most inner unless
65         end # of the less inner unless
66
67         @@nulla
68     end
69
70 @@ignota = nil
71 def self.ignota
72     unless @@ignota = self.find( :first, :conditions =>
73         { :klasa_gram_ozn => 'ign' } )
74         @@ignota = self.create!( {
75             :cz_m_id => nil,
76             :klasa_gram_nazwa => "ignota",
77             :klasa_gram_ozn => "ign",
78             :klasa_sem => nil
79         } )
80     end # of unless
81
82     @@ignota
83 end # of self.ignota
84
85 def self.kgs
86     unless defined?(@@kgs)
87         kgs = self.find(:all, :include => :cz_m)
88         @@kgs = Hash.new
89         kgs.each{|k|
90             @@kgs[k.klasa_gram_id] = [
91                 k.klasa_gram_ozn,
92                 if k.cz_m : k.cz_m.cz_m_nazwa ; else ""
93                 end
94             ]
95         }
96     end
97 end

```

```

94     end
95     return @@kgs
96   end # of self.kgs
99   end # of class

```

### **komentarz.rb**

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: komentarz
5   #
6   # komentarz_id :integer primary key
7   # akapit_id :integer
8   # uzytkownik_id :integer
9   # akapit_transzy_id :integer
10  # przyczyna_odrzucenia :integer default(o)
11  # tresc :text
12  # nowy :boolean default(TRUE)
13  # created_at :timestamp
14  # updated_at :timestamp
15  #

39  class Komentarz < ActiveRecord::Base
40    belongs_to :akapit
41    belongs_to :uzytkownik
42    belongs_to :akapit_transzy

45    PO = {
46      1 => "Błędne granice zdania",
47      2 => "Błędny opis morfologiczny",
48      3 => "Fraza nieciągła i synh &#8800; semh",
49      4 => "Błędna klas. fleksyjna czasownika z transzy",
50      5 => "Inne",
51      0 => "" }

53    PO_select = [ # ["(zd. OK)", o],
54                  ["złe granice zdania", 1],
55                  ["zły opis morfologiczny", 2],
56                  ["fraza nieciągła, synh semh", 3],
57                  ["zła klasyfikacja czasownika", 4],
58                  ["inne", 5] ]

61    def przyczyna
62      PO[self.przyczyna_odrzucenia]
63    end

66    def self.ile
67      self.find_by_sql( "select count(*) as ile from
68                        komentarz")[0].ile.to_i
69    end # of self.ile

71    def self.ile_akapitów

```

```
72     self.find_by_sql( "select count( distinct akapit_id ) as ile from
      komentarz")[0].ile.to_i
73   end # of self.ile
76   end
79   ## # Local Variables:
80   ## # mode: ruby
81   ## # End:
```

### **koniec\_zdania\_anot.rb**

```
1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: koniec_zdania_anot
5   #
6   # koniec_zdania_anot_id :integer primary key
7   # akapit_id :integer not null
8   # akapit_transzy_id :integer not null
9   # uzytkownik_id :integer not null
10  # token_id :integer not null
11  # created_at :timestamp
12  # updated_at :timestamp
13  #
37  class KoniecZdaniaAnot < ActiveRecord::Base
38    belongs_to :akapit_transzy
39    belongs_to :akapit
40    belongs_to :token
42    def self.znajdz( akat, token )
43      self.find( :first, :conditions => {
44        :akapit_transzy_id => akat,
45        :token_id => token
46      })
47    end # of self.znajdz
50    def bliźniaczy
51      @bli_hasz = {
52        :akapit_id => self.akapit_id,
53        :akapit_transzy_id => self.akapit_transzy.bliźniaczy_id,
54        :token_id => self.token_id
55      }
56      self.class.find( :first, :conditions => @bli_hasz )
57    end # of bliźniaczy
60    def zniszcz_oba
61      if bli = self.bliźniaczy : bli.destroy
62      end
63      self.destroy
64    end
66    def self.zrób_oba( hasz )
```

```

67     hasz_inicjalny = {
68       :akapit_id => hasz[:akat].akapit_id,
69       :akapit_transzy_id => hasz[:akat].akapit_transzy_id,
70       :uzytkownik_id => hasz[:uzytkownik_id],
71       :token_id => hasz[:token_id]}
72
73     unless self.znajdz( hasz[:akat], hasz[:token_id] )
74       self.create( hasz_inicjalny )
75     end
76     unless self.znajdz( ( abli = hasz[:akat].blizniaczy_id ) ,
77       hasz[:token_id] )
78       self.create(
79         hasz_inicjalny.update({ :akapit_transzy_id => abli })
80       )
81     end
82   end # of self.zrób_oba
83
84 end

```

### leksem.rb

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: leksem
5   #
6   # leksem_id :integer primary key
7   # lex_xmlid :text not null
8   # path_id :integer not null
9   # lemat :text
10  # klasa_gram_id :integer
11  # cz_m_id :integer
12  # cz_m_leksem_id :integer
13  # created_at :timestamp
14  # updated_at :timestamp
15  # cz_m_przypisana :boolean
16  # cz_m_leksem_przypisany :boolean
17  #
18
41  # to jest tabela pierwszych instancji danego lex-a w poszczególnych plikach
42  # ann_....
43  # lex_xmlid powinien być brany z tabeli interpretacja (i tak jest w exml.rb).
44  # w zasadzie należałoby utworzyć pośrednią tabelę lex między leksem a
45  # interpretacja.
46  # Miałaby ona unikalne lex_xmlid-y w obrębie danej ścieżki.
47
48  require 'ramkowanie'
49
50  class Leksem < ActiveRecord::Base
51    belongs_to :klasa_gram
52
53    belongs_to :cz_m_leksem
54    belongs_to :cz_m
55
56    def sensy_sensemu

```

```

56      # Raczej nierelevantne – w NKJP operujemy zawsze częścią mowy.
57      SensySensemu.find(:all,
58                          :conditions =>{
59                          :lemat => self.lemat, :klasa_sem =>
60                          self.klasa_gram.klasa_sem})
61  end# of sensy_sensemu

62  # Zdarzają się tokeny bez żadnej interpretacji, a zd
63  @@nullus = nil

64  def self.nullus
65      unless @@nullus
66          unless @@nullus = self.find( :first, :conditions => { :lemat =>
67              " } )
68              @@nullus = self.create( {
69                  :lex_xmlid => '0.0.0',
70                  :path_id => Path.dummy,
71                  :lemat => "",
72                  :klasa_gram_id => KlasaGram.nullus
73              } )
74          end # of the most inner unless
75          end # of the less inner unless
76          @@nullus
77      end

78  @@nulli_et_ignoti = nil

79  def self.nulli_et_ignoti
80      unless @@nulli_et_ignoti
81          # nie występuje problem odświeżania, bo po właniu danych do bazy
82          # nowe 'ign'-y mają nie móc być dodane.
83          @@nulli_et_ignoti = self.find( :all,
84              :conditions => " klasa_gram_id in
85              ({KlasaGram.nullus.id},
86              #{KlasaGram.ignota.id})" )
87      end
88      @@nulli_et_ignoti
89  end # of self.nulli_et_ignoti

90  @@inei = nulli_et_ignoti.collect{ |inus| inus.id }

91  def self.idi_nulli_et_ignoti
92      @@inei
93  end # of self.idi_nulli_et_ignoti

94  end # of class

```

## morph.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: morph
5  #

```

```

6   # morph_id :integer primary key
7   # xmlid :text not null
8   # path_text :text not null
9   # created_at :timestamp
10  # updated_at :timestamp
11  #

35  class Morph < ActiveRecord::Base
36    belongs_to :path, :foreign_key => "path_text"
37  end

```

### **morphosyntactic\_rozbieznosc.rb**

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: morphosyntactic_rozbieznosc
5   #
6   # morphosyntactic_rozbieznosc_id :integer primary key
7   # token_id :integer not null
8   # interpretacja_id :integer not null
9   # reszta_tagu :text
10  # leksem_id :integer not null
11  # created_at :timestamp
12  # updated_at :timestamp
13  #

37  class MorphosyntacticRozbieznosc < ActiveRecord::Base
38    has_one :punkt_protokolu, :as => :elt_protokol
39    belongs_to :token
40    belongs_to :interpretacja
41    belongs_to :leksem
42  end

45  ## # Local Variables:
46  ## # mode: ruby
47  ## # End:

```

### **nowa\_segmentacja.rb**

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: nowa_segmentacja
5   #
6   # nowa_segmentacja_id :integer primary key
7   # ids :text
8   # total_akapitow :integer
9   # seg_is :text
10  # seg_should_be :text
11  # nps :boolean default(TRUE), not null
12  # wprowadzona :boolean
13  # akapit_transzy_id :integer

```

```

14   # transza_id :integer
15   #

39   require 'ramkowanie'

42   class NowaSegmentacja < ActiveRecord::Base

44     has_one :prosby_annotatorek, :as => :dotyczy
45     serialize :ids # są to id-y akapitów zawierających układ tokenów
                     # podany w seg_is. Zapisujemy je w tabeli, żeby nie powtarzać kosztownego
                     # wyszukiwania, jednak użyjemy tej zapisanej wartości tylko wówczas, gdy
                     # w momencie jej użycia liczba akapitów w bazie jest taka, jak w momencie
                     # utworzenia danego rekordu.
46     serialize :seg_is
47     serialize :seg_should_be

49     include XpointerCreator

52     def fresh_ids
53       if self.total_akapitow == Akapit.count
54         self.ids
55       else
56         Akapit.znajdź_po_orthach( self.seg_is, :strict ).collect{ |a142|
          a142.id }
57       end
58     end # of fresh_ids

61     def instanze( is_or_should = :seg_is ) ## ( która=nil )
62       ins142 = []
63       is142 = self.send( is_or_should )
64       is_size142 = is142.size
65       # rozważałem utworzenie wersji tekstu akapitu, w której tokeny są
66       # rozdzielone swoimi id-ami i dopasowywanie specjalnie skonstruowanego
67       # wyrażenia regularnego. Ale to by było chyba kosztowniejsze od
68       # przejrzenia n-elementowych podtablic i porównanie ich równościowe.

69       ##      if która # a jednak wariant tablicowy w bazie III testu
70       ##      wykonuje się 7 s.
71       ##      Akapit.find( self.fresh_ids, :include => :token ).each
72       ##      { |a|
73       ##        0.upto( a.token.size - is_size ) { |i|
74       ##          if ( a.tokorths[ i, is_size ] ) == is
75       ##            ins « a.tokids[ i, is_size ]
76       ##          end
77       ##        }
78       ##      }
79       ##      else # a wariant regexpowy – w < 1 s.
80       rx142 = Regexp.new( is142.collect { |s142| Regexp.escape( s142 )
81       + '\\((\\d+)\\)' }.join )
82       ## puts rx142.inspect
83       subs_a142 = Proc.new { eval( '['
84       + ("$1".."$#{is142.size}") .to_a.join( ', ' ) + ']' ) }

```

```

88      # w poprzedniej wersji regexp zaczynało się od jeszcze jednej grupy
      '(?:^|/.*\d\))', co wobec zachłanności dopasowań powodowało
      znalezienie jedynie ostatniego wystąpienia szukanego układu tokenów.

90      Akapit.find( self.ids # było: fresh_ids, ale okazuje się, że należy to
      dodawać tylko w tym akapicie, w którym zgłoszono.

91      ).each {|a143|
92          ati143 = a143.treść_idowana
93          ## puts ati143
95      ati143.gsub( rx142 ) { |match| ins142 « subs_a142.call }
96      }

98      logger.info "#### instancje n.segm.: #{ins142.inspect}"

100     ins142.collect! {|ids142| Token.find( ids142 ) }

102     ## end
104     ins142
105 end # of instance

108 def should_xp( dii144 )
109     create_xpointers( self.seg_should_be, dii144, self.seg_is )
110     # z modułu XpointerCreator zdef. w ramkowanie.rb
111 end # of should_xp

114 def commit # wołana wyłącznie z aProsbyAnotatorek.commit.
115     # zwrócimy hasz idów obsługowanych akapitów i tokenów i dodanych
    tokenów
116     instantia1194 =
117         ## ( self.ids.sort == ( self.ids &
118             ## Akapit.znajdź_po_orthach(
119             self.seg_is, :strict ).collect{ |x194| x194.id }
120             ## ).sort )
121         self.instanze( :seg_is )[0] and # »jest« występuje
122         not self.instanze( :seg_should_be )[0] # »ma być« nie występuje

126     if ( not self.wprowadzona? ) and instantia1194
127         ## puts "#### nowa_segmentacja nie wprowadzona, popełniam"

130         di145 = self.instanze
131         ## puts " @@@ mam #{di145.size} wystąpień"
133         di_sgch_ids = di145.collect { |instanza|
134             ico145 = instanza.collect {|t145| t145.sg_choice_id
135             }.compact.uniq
136             raise "aNowaSegmentacja.commit: Obecnie nie obsługuję przypadku
137             przynależności do >1 sg_choice'ów" if ico145.size > 1
138             ico145[0]
139         }
140         akids145 = []
141         tokids145 = []
142         logger.info "#### aNowaSegmentacja.commit:
143         di_sgch_ids==#{di_sgch_ids.inspect}"
144         di145.each_index { |i145|
145             # dla każdej instancji układu tokenów »is«

```



```
143     dii145 = di145[ i145 ] # to są tokeny zastane
144     if j145 = di_sgch_ids[ i145 ]
145         sg_choice145 = SgChoice.find( j145 )
146     else
147         sg_choice145 = SgChoice.create!(
148             :akapit_id => dii145[ 0 ].akapit_id,
149             :dodany => true
150         )
151     end
152     sgvs145 = dii145.collect { |t146| t146.sg_variant }.compact

154     # jeśli mamy dokładnie 1 wariant segmentacyjny, to sprawdzamy, czy
    jest on równoważny naszemu »is«
155     # jeśli nie, to kopiujemy tokeny do nowego wariantu.
156     # jeśli jest więcej niż 1 wariant, to też kopiujemy tokeny.
157     if sgvs145.size == 1
158         if sgvs145[0].token.collect { |t147| t147.orth } ==
159             self.seg_is
160             # porównanie tabel tokenów wariantu i dii działało źle.
161             copy_is_variant = false
162             v145 = sgvs145[0]
163         else
164             copy_is_variant = true
165             v145 = nil
166         end
167     elsif sgvs145.size == 0
168         copy_is_variant = false
169         v145 = nil
170     else
171         copy_is_variant = true
172         v145 = nil
173     end

174     v145 ||= sg_choice145.sg_variant.create!( :dodany => true )

176     unless copy_is_variant
177         # not copy_is_variant
178         dii145.each { |t148|
179             t148.sg_choice_id = sg_choice145.id
180             t148.sg_variant_id = v145.id
181             t148.save!
182         }
183     end # of unless copy_is_variant

185     # Teraz tworzymy wariant(y) (ewentualnie dla »is« i) dla »should_be«.
    Że ten ostatni jest inny, sprawdziliśmy przy zapisywaniu prośby.

187     t_ost = dii145.collect { |t149|
188         ((( sv149 = t149.sg_variant ) && sv149.token ) || [t149] )[-1]
189         # jeśli dany token naszego układu ma sg_variant, to bierzemy
        ostatni token takiego wariantu, a jeśli dany token jest
        bezwariantowy, to bierzemy sam ten token. Z tak utworzonej tabeli
```

pobieramy element ostatni: jesteśmy pewni, że wszystkie tokeny danego wariantu następują przed następnym tokenem. Zapewne overkill, bo jest niemożliwe podanie jako »is« niespójnego podciągu tokenów (nie przeszłoby sprawdzenia).

```

190     }[-1]
192     logger.info "aNowaSegmentacja.commit 180"
194     kol_ost = t_ost.kolejnosc
195     raise "aNowaSegmentacja.commit: kolejność is NULL in token
      #{t_ost.id}" unless kol_ost
197     # token następny jest zawsze wybrany. A w przypadku dodawania
      wariantu segmentacyjnego nie zawsze musi o to chodzić: 2010/4/7
      zdarzyło się, że był już dodany jakiś wariant segmentacyjny, a chcieli
      dodać inny, lepszy. Powinni móc to zrobić, o ile tylko rozdzielczość
      kolejności na to pozwala.
198     ## kol_nast = if t178 = t_ost.następny( :nie_krzycz )
200     ## t178.kolejnosc
202     ## else kol_ost + 216
204     ## end
206     ## raise "aNowaSegmentacja.commit: kolejność is NULL in
      token #{t_ost.id}" unless kol_nast
208     kol_nast = Token.minimum( :kolejnosc, :conditions => "
      kolejnosc> #{kol_ost}" )
209     kol_nast = kol_ost + 216 unless kol_nast # w przypadku,
      gdybyśmy byli na kolejnościowym końcu tokenów.
211     news_size = if copy_is_variant
212     self.seg_is.size
213     else 0
214     end + self.seg_should_be.size
216     kol_nowa = ( kol_ost + kol_nast ) / 2
217     # przez chwilę zastanawiałem się, czy przedziału nie należy
      „wglębiać” o (mniejszą) połowę rozmiaru tego, co dodajemy. Myślę
      jednak, że nie: skutek „wglębiania” dostajemy po obu stronach coś
      być może nieparzystego, a bez wglębiania — co najwyżej po jednej.
219     if Token.find_by_kolejnosc( (kol_nowa...(kol_nowa
      + news_size)).to_a )
220     # to znajdzie nam pierwszy token o kolejności z podanego zakresu,
      o ile taki jest.
221     raise "aNowaSegmentacja.commit: próba zbyt zagnieżdżonego
      dodawania wariantów"
222     end
224     start = 0
226     copy_tokens = Proc.new { |field150, xp150|
227     ( ssb150 = field150 ).each_index{ |j150|
228     t150 = dii145[ 0 ]
229     t_nowy150 = Token.create!(
230     :kolejnosc => kol_nowa + start + j150,
```

```

231         :akapit_id => t150.akapit_id,
232         :xpointer => xp150[ j150 ],
233         ##      fs_morph_comment           :text
234         :path_id => t150.path_id,
235         :orth => ssb150[ j150 ],
236         :czy_interp => if /\w/ =~ ssb150[
237             j150 ] : false else true end,
238         :czy_konczy_zdanie => if j150 <
                ssb150.size - 1
239
240             false
241         else
242             dii145[ -1
243                 ].czy_konczy_zdanie?
244         end,
245         # jeśli tworzony token jest nieostatni, to
246         # nie kończy zdania (nie może być
247         # segmentacji międzyzdaniowej), jeśli zaś
248         # jest ostatni, to bierzemy wartość
249         # ostatniego tokenu zastanego.
250         :czy_konczy_zdanie_updated_at =>
251             Time.now,
252         :sg_choice_id => sg_choice145.id,
253         :sg_variant_id => v145.id,
254         :dodany => true,
255         :chosen => true
256         ##      chosen_updated_at
257         ##      created_at
258         ##      updated_at
259     )
260
261     ##      logger.info "=== committed token
262     ##      #{t_nowy150.inspect}"
263
264     xmlid150 = "#{t150.akapit_id}.#{t_nowy150.id}"
265
266     t_nowy150.ns_poprzedza = if j150 == 0
267         dii145[ 0 ].ns_poprzedza?
268     else
269         self.nps?
270     end
271
272     # jeśli jesteśmy pierwszym tokenem instancji, to bierzemy nps
273     # pierwszego zastanego; jeśli zaś nie, to bierzemy wartość
274     # przekazaną w prośbie.
275
276     t_nowy150.ns_nastepuje = if j150 < ssb150.size - 1
277         self.nps?
278     else
279         dii145[ -1 ].ns_nastepuje?
280     end
281
282     # dla nieostatniego tokenu bierzemy co podano w prośbie; dla
283     # ostatniego – wartość z ostatniego zastanego.
284
285     t_nowy150.segmentation_xmlid = xmlid150

```

```

277         t_nowy150.morphosyntactic_xmlid = xmlid150
278         t_nowy150.save!
279         akids145 « t150.akapit_id
280         tokids145 « t_nowy150.id
281     } # of each (is or) should-be token
282 } # of copy_token

284     start = 0
285     if copy_is_variant
286         copy_tokens.call( self.seg_is, dii145.collect {|t151|
287             t151.xpointer })
288         start = self.seg_is.size
289     end

290     # a teraz (wreszcie) ten nowy wariant.
291     v145 = sg_choice145.sg_variant.create!( :dodany => true )
292     copy_tokens.call( self.seg_should_be, self.should_xp( dii145 )
293 )

294 } # of each instanza (index)

296     if akids145[0]
297         Akapit.find( akids145.uniq, :include => :akapit_transzy ).each
298         { |a152|
299             a152.update_treści
300             a152.akapit_transzy.each{ |at152|
301                 at152.set_status( :segmentation, :dopuszczony, true )
302             }
303         }
304     end

305     self.wprowadzona = true
306     self.save!

308     if akids145[0] or tokids145[0]
309         { :akids => akids145, :tokids => tokids145 }
310     else
311         nil
312     end
313 elsif not instantial194
314     # jeżeli docelowy układ tokenów już występuje w którymś
315     # z podanych akapitów, to oznaczamy się jako wprowadzoną.
316     self.wprowadzona = true
317     self.save!
318 end # of if (not wprowadzona) and instantial194
319 logger.info "aNowaSegmentacja.commit 292"
320 end # of commit

321 end # of class

```

### path.rb

```

1  # == Schema Information
2  # Schema version: 51

```

```

3      #
4      # Table name: path
5      #
6      # path_id :integer primary key
7      # path_text :text not null
8      # created_at :timestamp
9      # updated_at :timestamp
10     #

34     class Path < ActiveRecord::Base
35       has_many :morph, :foreign_key => "path_text"
36       has_many :akapit

38       @@dummy = nil

40       def self.dummy
41         unless @@dummy
42           unless @@dummy = self.find( :first, :conditions => { :path_id =>
43             -1 } ) # bo find( -1) podnosi krzyk, gdy nie ma
44             self.connection.execute "insert into path( created_at,
45               path_id, path_text ) values( current_timestamp, -1,
46               'created_automatically_for_Interpretacja_and_Leksem' )"
47             end # of inner unless
48             end # of main unless
49             @@dummy ||= @@dummy = self.find( -1 )
50             # bo jeśli dummy jest, to drugie (wewn.) unless będzie fałszywe i zwróci
51             nil.
52             end # of self.dummy

53       def akapit_range
54         min199 = self.akapit.minimum( :akapit_id )
55         max199 = self.akapit.maximum( :akapit_id )
56         if min199 and max199
57           min199..max199
58         else
59           nil
60         end
61       end

63     end

```

## poziomy\_annotacji.rb

```

1      # == Schema Information
2      # Schema version: 51
3      #
4      # Table name: poziomy_annotacji
5      #
6      # poziomy_annotacji_id :integer primary key
7      # uzytkownik_id :integer not null
8      # segmentation :boolean
9      # sentences :boolean
10     # morphosyntactic :boolean

```

```

11 # word_senses :boolean
12 # syntactic_words :boolean
13 # named_entities :boolean
14 # syntactic :boolean
15 # created_at :timestamp
16 # updated_at :timestamp
17 #

41 # tabela przechowuje informacje, które poziomy anotacji są dostępne dla
    którego użytkownika. (Ale po co?)

43 require 'ramkowanie'

45 class PoziomyAnotacji < ActiveRecord::Base

47   belongs_to :uzytkownik

49   POZIOMY = [
50     # poziom, poprzedniki, następni
51     # kolejność – taka, w jakiej mają się wyświetlać na pasku.
52     [:segmentation, [].freeze, [:sentences,
    :morphosyntactic].freeze ].freeze,
53     [:sentences, [:segmentation].freeze,
    [:syntactic_words].freeze ].freeze,
54     [:morphosyntactic, [:segmentation].freeze, [:word_senses,
    :syntactic_words].freeze ].freeze,
55     [:word_senses, [:morphosyntactic].freeze, [].freeze
    ].freeze, # fakt, że WSD jest maksymalny, wykorzystuje
    Statusy.autozatwierdź_word_senses
56     [:syntactic_words, [:morphosyntactic, :sentences].freeze,
    [:named_entities, :syntactic].freeze ].freeze,
57     [:named_entities, [:syntactic_words].freeze, [].freeze
    ].freeze,
58     [:syntactic, [:syntactic_words].freeze, [].freeze ].freeze
59   ].freeze

62   STATI = {
63     :nie_dopuszczony => -1,
64     :dopuszczony => 0,
65     :zatwierdzony => 8, # jeśli Anotator zakończył (zaanotował lub
    odrzucił)
66     :do_poprawki => 10,
67     :do_konfrontacji => 10, # dla kompatybilności
68     :po_poprawce => 12,
69     :po_konfrontacji1 => 12, # dla kompatybilności
70     :po_konfrontacji2 => 12, # dla kompatybilności
71     :do_osądzania => 14, # jeśli zostało zatwierdzone i nie przeszło
    weryfikacji
72     :zweryfikowany => 16, # obaj Anotatorzy tak samo
73     :osądzany => 15, # było zweryfikowany, ale Zarządca coś w nim
    grzebnął, ale nie zatwierdził.
74     :osądzony => 18, # osądzony

```

```

75      :anotaŝne => [0, 10].freeze, # tzn. te, w kt3rych anotator na pewno
      ma coŝ do zrobienia
76      :oczekuj3ce => [8, 12].freeze # tzn. te, w kt3rych anotator czeka na
      partnera.
77  }.freeze

79  # uŝywane do zmiany statusu akapit, w kt3rym coŝ grzebiemy po jakimŝ
      etapie zatwierdzenia.
80  STATI_obniz = { -1 => -1, 0 => 0, 8 => 0, 10 => 10, 12=>10,
      14=>15, 15 => 15, 16 => 15, 18 => 15 }.freeze

82  STATI_podnies = {# anotacja_controller.podnies_status w wersji nk
83    # Po lewej stronie strzaŝki wyst3puj3 liczby lub tablice. Liczby – gdy
      podnosimy statusek bez weryfikacji, tablice – gdy podnosimy znaj3c
      wynik weryfikacji. Dlatego wŝasnie bez weryfikacji nie moŝna statusu
      podnieŝ do :zweryfikowany ani :os3dzony.
84    -1 => 0, # nie dopuszczony => dopuszczony
85    0 => 8, # dopuszczony (bez uwzgl3dnienia werdyktu) => zatwierdzony
86    [0, 0].freeze => 8, # dopuszczony i werdykt o (brak anotacji
      bliŝniaczej) => zatwierdzony
87    [0, 1].freeze => 16, # dopuszczony i werdykt 1 (zgodny)
88    [0, -1].freeze => 10, # dopuszczony i werdykt -1 (niezgodnoŝc)
89    8 => 8, # ŝeby nie krzyczaŝo.
90    [8, 0].freeze => 8, # zatwierdzony i jw. => zatwierdzony
91    [8, 1].freeze => 16, # zatwierdzony i jw. => zweryfikowany
92    [8, -1].freeze => 10, # zatwierdzony i jw. => do poprawki
93    10 => 12, # do poprawki => po poprawce
94    [10, 0].freeze => 12, # do poprawki i bd. => po poprawce
95    [10, 1].freeze => 16, # do poprawki i zdodne => zweryfikowany
96    [10, -1].freeze => 14, # do poprawki i niezgodne => do os3dzenia
97    12 => 12, # po poprawce bez wiedzy o wyniku weryfikacji => do
      os3dzenia
98    [12, 0].freeze => 12, # po poprawce i bd. => po poprawce (infinite
      loop?)
99    [12, 1].freeze => 16, # po poprawce i zgodne => zweryfikowany
100   [12, -1].freeze => 14, # po poprawce i niezgodne => do os3dzenia
101   14 => 15, # do os3dzenia => os3dzany
102   [14, -1].freeze => 18, # do os3dzenia => os3dzony
103   [14, 0].freeze => 18, # do os3dzenia => os3dzony
104   [14, 1].freeze => 18, # do os3dzenia => os3dzony
105   15 => 15,
106   [15, -1].freeze => 18, # os3dzany => os3dzony
107   [15, 0].freeze => 18, # os3dzany => os3dzony
108   [15, 1].freeze => 18, # os3dzany => os3dzony
109   18 => 18, # os3dzony => os3dzony (Arbiter moŝe zmieniŝ sw3j os3d)
110   [18, -1].freeze => 18, # os3dzony => os3dzony (Arbiter moŝe
      zmieniŝ sw3j os3d)
111   [18, 0].freeze => 18, # os3dzony => os3dzony (Arbiter moŝe zmieniŝ
      sw3j os3d)
112   [18, 1].freeze => 18 # os3dzony => os3dzony (Arbiter moŝe zmieniŝ
      sw3j os3d)

```

```

113     }.freeze

115     STATI_tekst = {
116       -1 => "nie dopuszczony do an.",
117       0 => "dopuszczony do an.",
118       8 => "zatwierdzony",
119       10 => "do poprawki",
120       12 => "po poprawce",
121       14 => "do osądzenia",
122       16 => "zweryfikowany",
123       15 => "osądzany",
124       18 => "osądzony"
125     }.freeze

127     STATI_kr = {
128       -1 => "&#8722;", # minus 8722, en dash 8211
129       0 => "dop",
130       8 => "zatw",
131       10 => "dopo",
132       12 => "popo",
133       14 => "doos",
134       16 => "zwer",
135       15 => "osa",
136       18 => "oso"
137     }.freeze

139     if true or AnoVersion.port == 8005
140       DZIAŁAJĄCE = [:segmentation, :sentences, :morphosyntactic,
141         :word_senses].freeze
142     else
143       DZIAŁAJĄCE = [:segmentation, :sentences, :morphosyntactic].freeze
144     end

146     def self.działająca
147       DZIAŁAJĄCE
148     end # of self.działająca

150     # Poziomy maksymalne to z definicji te, które nie mają następników.
151     # Poziomy minimalne – te, które nie mają poprzedników. Definiujemy
152     # zmienne klasowe @@poziomy_maksymalne i @@poziomy_minimalne, a także
153     # metody self.maksymalne i self.minimalne dające odczyt tych zmiennych.
154     tekscik = "    @@poziomy_xxx = POZIOMY.collect{ |poz|      poz[0]
155       unless poz[yyy][0]    }.compact.freeze    def self.xxx
156       @@poziomy_xxx    end    "
157     eval tekscik.gsub( 'xxx', 'maksymalne').gsub('yyy', '2')
158     eval tekscik.gsub( 'xxx', 'minimalne').gsub('yyy', '1')

159     @@poziomy_x = POZIOMY.dx

161     def self.poziomy_x
162       @@poziomy_x
163     end

164     NIE_DZIAŁAJĄCE = (@@poziomy_x - DZIAŁAJĄCE).freeze

```



```

163     def self.nie_działające
164         NIE_DZIAŁAJĄCE
165     end

167     def self.poprzedniki( poziom )
168         POZIOMY[ poziomy_x.index( poziom ) ][1] & DZIAŁAJĄCE
169     end

171     def self.nastepniki( poziom )
172         POZIOMY[ poziomy_x.index( poziom ) ][2] & DZIAŁAJĄCE
173     end

176     @@wyzsze = {}

178     def self.wyzsze( poziom )
179         # transitive closure operacji następnika.
180         unless @@wyzsze[ poziom ]
181             upper = []
182             succs = self.nastepniki( poziom )

184             until succs == []
185                 succs1 = []
186                 succs.each{ |succ|
187                     upper « succ
188                     succs1 « self.nastepniki( succ )
189                 }
190                 succs = succs1.flatten.uniq
191             end# of until
192             @@wyzsze[ poziom ] = upper.uniq
193         end
194         @@wyzsze[ poziom ]

196     end# of self.wyzsze.

199     @@poziomy_x.each{ |poz| self.wyzsze( poz )}

202     NAZWY = {
203         :segmentation => { :nom => 'segmentacja', :gen => 'segmentacji',
204             :kr => 'seg' }.freeze,
205         :sentences => { :nom => 'granice zdań', :gen => 'granic zdań', :kr
206             => 'sn' }.freeze,
207         :morphosyntactic => { :nom => 'morfoskładnia', :gen =>
208             'morfoskładni', :kr => 'ms' }.freeze,
209         :word_senses => { :nom => 'sensy słów', :gen => 'sensów słów', :kr
210             => 'wsd' }.freeze,
211         :syntactic_words => { :nom => 'słowa składniowe', :gen => 'słów
212             składniowych', :kr => 'synw' }.freeze,
213         :named_entities => { :nom => 'byty nazwane', :gen => 'bytów
214             nazwanych', :kr => 'nen' }.freeze,
215         :syntactic => { :nom => 'a. składniowa', :gen => 'a. składniowej',
216             :kr => 'syn' }.freeze
217     }.freeze

219     def self.krótkie( poziomy )

```

```

213     if poziomy.kind_of?( Symbol )
214       NAZWY[ poziomy ][:kr]
215     elsif poziomy.kind_of?( Array )
216       poziomy.collect { |po|
217         NAZWY[ po ][:kr] }
218     else
219       raise "PoziomyAnotacji.krótkie: type of the argument
220         (#{poziomy.inspect}) not supported."
221     end
222   end # of self.krótkie

223   tekscik =
224     "@@poziomy_xxx = {}    @@poziomy_x.each{ |poz|
225     @@poziomy_xxx[poz] = NAZWY[poz][:xxx]}.freeze    def self.xxx(
226     poziom )    @@poziomy_xxx[ poziom ]    end    @@all_xxx =
227     @@poziomy_x.collect{ |poz|    self.xxx( poz )}.freeze    def
228     self.all_xxx    @@all_xxx    end    @@działające_xxx =
229     DZIAŁAJĄCE.collect { |poz| self.xxx( poz )}.freeze    def
230     self.działające_xxx    @@działające_xxx    end    "

231   eval( tekscik.gsub( 'xxx', 'gen' ))
232   eval( tekscik.gsub( 'xxx', 'nom' ))
233   eval( tekscik.gsub( 'xxx', 'kr' ))
234   # to nam daje metody self.gen, self.nom i self.kr, wszystkie trzy z 1
235   argumentem <poziom>,
236   # oraz metody self.all_gen, self.all_nom i self.all_kr zwracające,
237   a także self.działające_nom itd.

238   def self.znajdz_u( uzid )
239     ich = self.find_by_uzytkownik_id( uzid )
240     unless ich
241       self.nadaż_za_uzytkownikiem
242       ich = self.find_by_uzytkownik_id( uzid )
243     end
244     ich
245   end

246   def widoczne
247     wi = []
248     @@poziomy_x.each { |poziom|
249       wi « poziom if self.send( "#{poziom}?" )
250     }
251     return wi
252   end # of widoczne

253   def veriﬁables( akat )
254     sensowne = []
255     as = akat.statuski
256     as.each_index{ |i|
257       sensowne « @@poziomy_x[i] if ( 8...16 ).include?( as[i] )
258     }
259     return sensowne

```

```

257     end # of verifiabiles

259     def anotables( akat, rola_id )
260       sensowne = []
261       if Rola.zarzadca?( rola_id )
262         maxim = 16
263       else
264         maxim = 14
265       end
266       as = akat.statuski
267       as.each_index{ |i|
268         sensowne « @@poziomy_x[i] if (0...maxim).include?( as[i] )
269       }
270       return sensowne
271   end# of anotables

273   def anotaśne( akat, rola_id )
274     sensowne = []
275     if Rola.zarzadca?( rola_id )
276       picipolo = 0...16
277     else
278       picipolo = STATI[ :anotaśne ]
279     end
280     as = akat.statuski
281     as.each_index{ |i|
282       sensowne « @@poziomy_x[i] if picipolo.include?( as[i] )
283     }
284     return sensowne
285   end# of anotaśne

287   ##      def pokażne( akat, rola_id )
288   ##          ano = self.anotables( akat, rola_id )
289   ##          wido = self.widoczne
290   ##          if ano & wido == []
291   ##              ( wido + ano ).uniq
292   ##          else wido
293   ##          end
294   ##      end # of pokażne

303   def z_pokażnymi( akat, rola_id )
304     z_poka = PoziomyAnotacji.new
305     coś_wybrano = false

307     self.attributes.each{ |key, val| # dup jest za słabe.
308       z_poka.send( "#{key}=", val # self.send( key )
309         ) unless "#{key}" == 'poziomy_annotacji_id'
310       coś_wybrano = true if self.send( key ).class == TrueClass
311       # korzystamy z tego, że tylko kolumny poziomów mogą przyjmować
       # wartości Boolowskie.
312     }

313     unless coś_wybrano
314       ano = self.anotaśne( akat, rola_id )

```

```

315         ano = DZIAŁAJĄCE unless ano[0]
316         ano.each{ |poz|
317             z_poka.send( "#{poz}=", true ) }
318         end

320         z_poka
321     end # of z_pokażnymi!

323     def colspan
324         # wi = widoczne
325         # cs = wi.size
326         # cs -= 1 if wi.include?( :segmentation )
327         # cs -= 1 if wi.include?( :word_senses ) sensy są w tej samej kolumnie
328         # cs += 1 if wi.include?( :syntactic ) składniowa jest w 2 kolumnach
329         cs = 1 # morfosyntaks i ew. sensy
330         cs += 2 if self.syntactic?
331         return cs
332     end

334     def set_poziom( poziom, wartosc )
335         self.send( "#{poziom}=", wartosc )
336     end

338     def trues
339         @@poziomy_x.collect { |poz|
340             if self.send( "#{poz}?" ) : poz
341             else nil
342             end}.
343         compact
344     end

348     def self.nadaż_za_użytkownikiem
349         Uzytkownik.natror # żeby zainicjować
350         self.connection.execute " insert into poziomy_annotacji(
351             created_at, uzytkownik_id ) " +
352             " select current_timestamp, u.uzytkownik_id from " +
353             " uzytkownik u left outer join poziomy_annotacji po " +
354             " on u.uzytkownik_id=po.uzytkownik_id " +
355             " where po.uzytkownik_id is null "

356         self.connection.execute " delete from poziomy_annotacji " +
357             " where uzytkownik_id in " +
358             " ( select po.uzytkownik_id from poziomy_annotacji po left outer
359             join uzytkownik u " +
360             " on u.uzytkownik_id=po.uzytkownik_id " +
361             " where u.uzytkownik_id is null ) "
362         true
363     end # of self.nadaż_za_użytkownikiem

365     def self.tylko_działające
366         niedziałające = NIE_DZIAŁAJĄCE
367         i=0

```

```

368         self.find( :all ).each{ |po|
369             niedziałające.each{ |nd|
370                 if po.send( "#{nd}?" )
371                     po.send( "#{nd}=", false )
372                     po.save!
373                     i += 1
374                 end
375             }
376         }
377         return i
378     end # of tylko_działające
381 end
384 ## # Local Variables:
385 ## # mode: ruby
386 ## # End:

```

### prosby\_annotorek.rb

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: prosby_annotorek
5   #
6   # prosby_annotorek_id :integer primary key
7   # dotyczy_id :integer not null
8   # dotyczy_type :text not null
9   # uzytkownik_id :integer not null
10  # opis :text
11  # created_at :timestamp
12  # updated_at :timestamp
13  # rozpatrzona :boolean
14  # spelniona :boolean
15  #
39  class ProsbyAnotorek < ActiveRecord::Base
40      belongs_to :uzytkownik
41      belongs_to :dotyczy, :polymorphic => true
43      serialize :ruby_data
45      def self.nowe( rola_id )
46          conds = { :rozpatrzona => false }
47          unless Rola.zarzadca?( rola_id )
48              conds[ :dotyczy_type ] = "NowaSegmentacja"
49          end
51          self.find(:all,
52                  :include => :uzytkownik,
53                  :order => "#{self.table_name}.created_at",
54                  :conditions => conds)
55      end # of self.nowe

```

```

58 K_segmentacyjna = "Czy rzeczywiście chcesz dodać nowy wariant
segmentacyjny?\nSpowoduje to w szczególności cofnięcie odp.
akapitu/ów na najniższy poziom anotacji."

60 K_transzowa = "Czy rzeczywiście chcesz odebrać transzę?"

62 def rodzaj
63   if self.dotyczy_type == "Transza"
64     "odebranie transzy"
65   else
66     "segmentacja"
67   end
68 end # of rodzaj

70 def self.k_segmentacyjna
71   K_segmentacyjna
72 end

74 def konfirmacja
75   if self.dotyczy_type == "Transza"
76     K_transzowa
77   else
78     K_segmentacyjna
79   end
80 end # of konfirmacja

82 def treść_acc # do „odrzuć/spełnił prośbę o ...”
83   dot = self.dotyczy
84   if dot.is_a?( Transza )
85     "odebranie transzy #{dot.id}"
86   elsif dot.is_a?( NowaSegmentacja )
87     seg_is = dot.seg_is
88     seg_should_be = dot.seg_should_be
89     zespa, zespójnik = if dot.nps? : [ "(z brakiem spacji)", "/" ]
90                       else [ "(ze spacjami)", " " ]
91                       end
92     zespa = nil      if seg_should_be.size == 1

94     "nowy wariant segmentacyjny »<b>#{seg_is.join(' ')}</b>« &#8614;
»<b>#{seg_should_be.join( zespójnik )}</b>« #{zespa} w
akapicie/tach <b>#{dot.ids.join(', ')}</b>" +
95     if akat = dot.akapit_transzy_id : " (dla akatru #{akat}
w&nbsp;transzy <b>#{dot.transza_id}</b>)" else " end
96   end
97 end # of treść_acc

100 def treść_z_opisem
101   treść_acc.gsub( /\zmianę segmentacji/, 'zmiana segmentacji') +
102   if o = self.opis : " <br/> #{o}" else " end
103 end # of treść_z_opisem

105 def odpfajkuj( wynik = true )
106   self.class.transaction do
107     self.rozpatrzone = true

```

```

108         self.spelniona = wynik
109         self.save!
110     end # of transaction
111 end # of odfajkuj

114 def commit( rola_id )
115     if self.dotyczy.is_a?( Transza ) and Rola.zarzadca?( rola_id )

117         self.class.transaction do
118             self.dotyczy uzytkownik_id = nil
119             self.dotyczy.save!
120             self.odfajkuj
121         end # of transaction
122         "Odebrałaś/eś #{dotyczy.uzytkownik.login} transzę
        #{dotyczy.opis}."

124     elsif self.dotyczy.is_a?( NowaSegmentacja ) and # wariant
        segmentacyjny
125         Rola.wszewid?( rola_id )
126         self.class.transaction do
127             self.dotyczy.commit
128             self.odfajkuj
129         end # of transaction
130         "Dodałeś #{self.treść_acc}."
131     else
132         "Nie masz prawa spełnić tej prośby."
133     end # of rodzaje próśb
134 end # of commit

137 def odrzuć( rola_id )
138     if Rola.zarzadca?( rola_id ) or (
139         self.dotyczy.is_a?( NowaSegmentacja )
140         and # wariant segmentacyjny
        Rola.wszewid?( rola_id ) )

142         self.odfajkuj( false )
143         "Odrzuciłeś prośbę #{self.uzytkownik.login} o
        #{self.treść_acc}."
144     else
145         "Nie masz prawa spełnić tej prośby."
146     end #
147 end # of odrzuć

150 end # of class

```

## protokol.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: protokol
5  #
6  # protokol_id :integer primary key
7  # poziom :text not null

```

```

8   # akapit_id :integer not null
9   # czy_rozbieznosc :boolean default(TRUE), not null
10  # czy_superancja :boolean not null
11  # created_at :timestamp
12  # updated_at :timestamp
13  #

37  class Protokol < ActiveRecord::Base

39  # Uwaga. Są w bazie rekordy bez powiązanych punktów protokołu. Znaczy
    to, że weryfikacja, której dotyczą, przebiegła pomyślnie (nie było żadnej
    rozbieżności do zaprotokołowania).

41    has_many :punkt_protokołu,
42      :order => "elt_protokol_type, elt_protokol_id",
43      :dependent => :delete_all
44    serialize :poziom

46    def self.zrób_się( akapit, poziom )
47      self.create(
48        :poziom => poziom,
49        :akapit_id => akapit.id,
50        :czy_rozbieznosc => true
51      )
52    end # of self.zrób_się

54    def build_ppr( akat, uzid=nil )
55      self.punkt_protokołu.build(
56        :akapit_transzy_id => akat.id,
57        :uzytkownik_id => ( uzid || akat.get_uzid )
58      )
59    end # of build_ppr

60    def protokoluj_segmentation( akat, sg_choice, sg_variant, uzid=nil )
61      ppr = build_ppr( akat, uzid )
62      ppr.elt_protokol = SegmentationRozbieznosc.new(
63        :sg_choice_id =>
64          sg_choice.id,
65        :sg_variant_id =>
66          sg_variant.id
67      )
68      ppr.save!
69    end # of protokoluj_segmentation.

70    def protokoluj_sentences( akat, tok, uzid=nil )
71      ppr = build_ppr( akat, uzid )
72      ppr.elt_protokol = SentencesRozbieznosc.new(
73        :token_id => tok.id,
74        :czy_konczy_zdanie => (
75          if
            KoniecZdani-
            aAnot.znajdz(
              akat, tok
            )
          )
76      )
77    end # of protokoluj_sentences.

```



```

76                                     true
77                                     else false
78                                     end
79                                     )
80                                     )
81     ppr.save!
82 end # of protokoluj_sentences
83
84 def protokoluj_morphosyntactic( akat, tok, disa, uzid=nil )
85     ppr = build_ppr( akat, uzid )
86
87     if disa
88         ppr.elt_protokol = MorphosyntacticRozbieznosc.new(
89             :token_id =>
90                 disa.token_id,
91             :interpretacja_id =>
92                 disa.id,
93             :reszta_tagu =>
94                 disa.reszta_tagu,
95             :leksem_id =>
96                 disa.leksem_id)
97
98     else
99         ppr.elt_protokol_nil
100         logger.info "#### null disambiguation for token »#{tok.orth}«
101             #{tok.id} in akapit_transzy #{akat.id}"
102     end
103
104     ppr.save!
105 end # of self.protokoluj_morphosyntactic.
106
107 def protokoluj_word_senses( akat, tok, sens, uzid=nil )
108     ppr = build_ppr( akat, uzid )
109     ppr.elt_protokol = WordSensesRozbieznosc.new(
110         :token_id => tok.id,
111         :sensy_id => sens.id
112     )
113
114     ppr.save!
115 end # of protokoluj_sentences
116
117 def self.superancje( poziom, akat, uzid )
118     ak = akat.akapit
119
120     v = ak.superancje[ poziom ]
121     # dostajemy hasz indeksowany poziomami, którego wartości to tablice
122     tokenów
123     if v
124         pr = self.create(
125             :poziom => poziom,
126             :akapit_id => ak.id,
127             :czy_rozbieznosc => false,
128             :czy_superancja => true
129         )
130
131         case poziom
132         when 1
133             pr.elt_protokol = WordSensesRozbieznosc.new(
134                 :token_id => v[0].id,
135                 :sensy_id => v[0].sensy_id,
136                 :poziom => poziom,
137                 :akapit_id => ak.id,
138                 :czy_rozbieznosc => false,
139                 :czy_superancja => true
140             )
141         when 2
142             pr.elt_protokol = WordSensesRozbieznosc.new(
143                 :token_id => v[0].id,
144                 :sensy_id => v[0].sensy_id,
145                 :poziom => poziom,
146                 :akapit_id => ak.id,
147                 :czy_rozbieznosc => false,
148                 :czy_superancja => true
149             )
150         when 3
151             pr.elt_protokol = WordSensesRozbieznosc.new(
152                 :token_id => v[0].id,
153                 :sensy_id => v[0].sensy_id,
154                 :poziom => poziom,
155                 :akapit_id => ak.id,
156                 :czy_rozbieznosc => false,
157                 :czy_superancja => true
158             )
159         when 4
160             pr.elt_protokol = WordSensesRozbieznosc.new(
161                 :token_id => v[0].id,
162                 :sensy_id => v[0].sensy_id,
163                 :poziom => poziom,
164                 :akapit_id => ak.id,
165                 :czy_rozbieznosc => false,
166                 :czy_superancja => true
167             )
168         when 5
169             pr.elt_protokol = WordSensesRozbieznosc.new(
170                 :token_id => v[0].id,
171                 :sensy_id => v[0].sensy_id,
172                 :poziom => poziom,
173                 :akapit_id => ak.id,
174                 :czy_rozbieznosc => false,
175                 :czy_superancja => true
176             )
177         when 6
178             pr.elt_protokol = WordSensesRozbieznosc.new(
179                 :token_id => v[0].id,
180                 :sensy_id => v[0].sensy_id,
181                 :poziom => poziom,
182                 :akapit_id => ak.id,
183                 :czy_rozbieznosc => false,
184                 :czy_superancja => true
185             )
186         when 7
187             pr.elt_protokol = WordSensesRozbieznosc.new(
188                 :token_id => v[0].id,
189                 :sensy_id => v[0].sensy_id,
190                 :poziom => poziom,
191                 :akapit_id => ak.id,
192                 :czy_rozbieznosc => false,
193                 :czy_superancja => true
194             )
195         when 8
196             pr.elt_protokol = WordSensesRozbieznosc.new(
197                 :token_id => v[0].id,
198                 :sensy_id => v[0].sensy_id,
199                 :poziom => poziom,
200                 :akapit_id => ak.id,
201                 :czy_rozbieznosc => false,
202                 :czy_superancja => true
203             )
204         when 9
205             pr.elt_protokol = WordSensesRozbieznosc.new(
206                 :token_id => v[0].id,
207                 :sensy_id => v[0].sensy_id,
208                 :poziom => poziom,
209                 :akapit_id => ak.id,
210                 :czy_rozbieznosc => false,
211                 :czy_superancja => true
212             )
213         when 10
214             pr.elt_protokol = WordSensesRozbieznosc.new(
215                 :token_id => v[0].id,
216                 :sensy_id => v[0].sensy_id,
217                 :poziom => poziom,
218                 :akapit_id => ak.id,
219                 :czy_rozbieznosc => false,
220                 :czy_superancja => true
221             )
222         when 11
223             pr.elt_protokol = WordSensesRozbieznosc.new(
224                 :token_id => v[0].id,
225                 :sensy_id => v[0].sensy_id,
226                 :poziom => poziom,
227                 :akapit_id => ak.id,
228                 :czy_rozbieznosc => false,
229                 :czy_superancja => true
230             )
231         when 12
232             pr.elt_protokol = WordSensesRozbieznosc.new(
233                 :token_id => v[0].id,
234                 :sensy_id => v[0].sensy_id,
235                 :poziom => poziom,
236                 :akapit_id => ak.id,
237                 :czy_rozbieznosc => false,
238                 :czy_superancja => true
239             )
240         when 13
241             pr.elt_protokol = WordSensesRozbieznosc.new(
242                 :token_id => v[0].id,
243                 :sensy_id => v[0].sensy_id,
244                 :poziom => poziom,
245                 :akapit_id => ak.id,
246                 :czy_rozbieznosc => false,
247                 :czy_superancja => true
248             )
249         when 14
250             pr.elt_protokol = WordSensesRozbieznosc.new(
251                 :token_id => v[0].id,
252                 :sensy_id => v[0].sensy_id,
253                 :poziom => poziom,
254                 :akapit_id => ak.id,
255                 :czy_rozbieznosc => false,
256                 :czy_superancja => true
257             )
258         when 15
259             pr.elt_protokol = WordSensesRozbieznosc.new(
260                 :token_id => v[0].id,
261                 :sensy_id => v[0].sensy_id,
262                 :poziom => poziom,
263                 :akapit_id => ak.id,
264                 :czy_rozbieznosc => false,
265                 :czy_superancja => true
266             )
267         when 16
268             pr.elt_protokol = WordSensesRozbieznosc.new(
269                 :token_id => v[0].id,
270                 :sensy_id => v[0].sensy_id,
271                 :poziom => poziom,
272                 :akapit_id => ak.id,
273                 :czy_rozbieznosc => false,
274                 :czy_superancja => true
275             )
276         when 17
277             pr.elt_protokol = WordSensesRozbieznosc.new(
278                 :token_id => v[0].id,
279                 :sensy_id => v[0].sensy_id,
280                 :poziom => poziom,
281                 :akapit_id => ak.id,
282                 :czy_rozbieznosc => false,
283                 :czy_superancja => true
284             )
285         when 18
286             pr.elt_protokol = WordSensesRozbieznosc.new(
287                 :token_id => v[0].id,
288                 :sensy_id => v[0].sensy_id,
289                 :poziom => poziom,
290                 :akapit_id => ak.id,
291                 :czy_rozbieznosc => false,
292                 :czy_superancja => true
293             )
294         when 19
295             pr.elt_protokol = WordSensesRozbieznosc.new(
296                 :token_id => v[0].id,
297                 :sensy_id => v[0].sensy_id,
298                 :poziom => poziom,
299                 :akapit_id => ak.id,
300                 :czy_rozbieznosc => false,
301                 :czy_superancja => true
302             )
303         when 20
304             pr.elt_protokol = WordSensesRozbieznosc.new(
305                 :token_id => v[0].id,
306                 :sensy_id => v[0].sensy_id,
307                 :poziom => poziom,
308                 :akapit_id => ak.id,
309                 :czy_rozbieznosc => false,
310                 :czy_superancja => true
311             )
312         when 21
313             pr.elt_protokol = WordSensesRozbieznosc.new(
314                 :token_id => v[0].id,
315                 :sensy_id => v[0].sensy_id,
316                 :poziom => poziom,
317                 :akapit_id => ak.id,
318                 :czy_rozbieznosc => false,
319                 :czy_superancja => true
320             )
321         when 22
322             pr.elt_protokol = WordSensesRozbieznosc.new(
323                 :token_id => v[0].id,
324                 :sensy_id => v[0].sensy_id,
325                 :poziom => poziom,
326                 :akapit_id => ak.id,
327                 :czy_rozbieznosc => false,
328                 :czy_superancja => true
329             )
330         when 23
331             pr.elt_protokol = WordSensesRozbieznosc.new(
332                 :token_id => v[0].id,
333                 :sensy_id => v[0].sensy_id,
334                 :poziom => poziom,
335                 :akapit_id => ak.id,
336                 :czy_rozbieznosc => false,
337                 :czy_superancja => true
338             )
339         when 24
340             pr.elt_protokol = WordSensesRozbieznosc.new(
341                 :token_id => v[0].id,
342                 :sensy_id => v[0].sensy_id,
343                 :poziom => poziom,
344                 :akapit_id => ak.id,
345                 :czy_rozbieznosc => false,
346                 :czy_superancja => true
347             )
348         when 25
349             pr.elt_protokol = WordSensesRozbieznosc.new(
350                 :token_id => v[0].id,
351                 :sensy_id => v[0].sensy_id,
352                 :poziom => poziom,
353                 :akapit_id => ak.id,
354                 :czy_rozbieznosc => false,
355                 :czy_superancja => true
356             )
357         when 26
358             pr.elt_protokol = WordSensesRozbieznosc.new(
359                 :token_id => v[0].id,
360                 :sensy_id => v[0].sensy_id,
361                 :poziom => poziom,
362                 :akapit_id => ak.id,
363                 :czy_rozbieznosc => false,
364                 :czy_superancja => true
365             )
366         when 27
367             pr.
```

```

130      when :segmentation
131          ##                                sg_choice, sg_variant, uzid=nil
132          params = v.collect { |tok|
133              [tok.sg_choice_id, tok.sg_variant_id]          if tok.chosen?
134          }.compact.uniq
135          params.each{ |par|
136              pr.protokoluj_segmentation( akat, par[0], par[1], uzid )
137          }
138      when :sentences
139          # :sentences: akat, tok, uzid=nil
140          v.each{ |tok|          pr.protokoluj_sentences( akat, tok, uzid
141              ) }
142
143      when :morphosyntactic
144          # :morphosyntactic: akat, tok, disa, uzid=nil
145          v.each{ |tok|
146              pr.protokoluj_morphosyntactic(
147                  akat, tok, tok.disambs(
148                      akat.akat_hash )[0], uzid )
149          }
150      when :word_senses
151          v.each{ |tok|
152              pr.protokoluj_word_senses(
153                  akat, tok, tok.sensy( akat.akat_hash
154                      )[0], uzid )
155          }
156      else
157          raise "Protokol.superancje: unhandled poziom #{poziom.inspect}"
158      end
159  end # of if v
160
161  end # of superancje
162
163  end# of class
164
165  ## # Local Variables:
166  ## # mode: ruby
167  ## # End:

```

### punkt\_protokolu.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: punkt_protokolu
5  #
6  # punkt_protokolu_id :integer primary key
7  # elt_protokol_id :integer not null
8  # elt_protokol_type :text not null
9  # protokol_id :integer not null
10 # akapit_transzy_id :integer not null
11 # uzytkownik_id :integer

```

```

12 # created_at :timestamp
13 # updated_at :timestamp
14 #

38 class PunktProtokolu < ActiveRecord::Base
39   belongs_to :elt_protokol, :polymorphic => true
40   # mówi to, że kolumny elt_protokol_id i elt_protokol_type
41   # tworzą polimorficzny klucz obcy.

43   def elt_protokol_nil
44     self.elt_protokol_type = "NilClass"
45     self.elt_protokol_id= 0
46   end

49 end

52 ## # Local Variables:
53 ## # mode: ruby
54 ## # End:

```

### ramkowanie.rb

```

23 module Ramkowanie
24   # moduł, który włączam do FrazaAnot i do Ramka.

26   CASES = /(nom|gen|dat|acc|inst|loc|voc)/

28   UNDEF = 'UNDEF'

31   @@srt = nil

33   def self.srt
34     unless @@srt
35       @@srt = Proc.new {|a128, b128|
36         x134 = (a128[0] <=> b128[0])
37         x134= (a128[1] <=> b128[1] ) if x134 == 0
38         x134}
39     end
40     @@srt
41   end # of self.srt.

43   end # of module Ramkowanie.

46   class Ihash < Hash
47     def self.new( a_hash134={} )
48       h130 = Hash.new {
49         # Proca, która będzie wykonywana dla domyślnego eltu hasza.
50         # Służy znalezieniu odp. wartości w haszu indeksowanym
           przedziałami lub tablicami.
51         |hash129, key129|
52         found134 = false
53         v130 = hash129.each{|k129, val129|
54           if k129.respond_to?( :include? ) and k129.include?( key129 )
55             found134 = true
56             break( val129 )

```

```

57         end}
58         v130 if found134
59     }
60     h130.merge!( a_hash134 )
61     return h130
62 end# of self.new,

64 end # of class Ihash

69 class Array

71     def proj( axis131 )
72         self.collect{ |x131| x131[axis131] }
73     end

75     def dx
76         self.proj( 0 )
77     end

79     def dy
80         self.proj( 1 )
81     end

83     def dz
84         self.proj( 2 )
85     end

87     def dt
88         self.proj( 3 )
89     end

92     def includetext? ( tx132 )
93         # zadziała dla tablicy stringów, i zwróci false, jeśli żaden ze stringów
94         # nie zawiera/regexp tx, albo tablicę indeksów dopasowania.
95         incl133 = Array.new
96         if tx132.kind_of?( Regexp )
97             self.each_index{ |i132|
98                 incl133 « i132 if (self[i132] =~ tx132) }
99             elsif tx132.kind_of?( String )
100                 self.each_index{ |i133|
101                     incl133 « i133 if self[i133].include?( tx132 ) }
102             end
103             incl133 = false unless incl133[ 0 ]
104             return incl133
105         end # of includetext?

108     def to_arghash( ohash135 = {} )
109         # zakładamy, że jest to tablica [ tag, sensy-opis, ew. vp_typ lub typ_sie ]
110         h135=self[0].to_arghash( ohash135 )
111         s135=self[1].split.collect{|s136| Sensy.opis2ozn(s136) }

113         h135[:sens] = s135 if s135[0]

115         if h135[:syncat] == FrazaTyp.downhash['sie'] and self[2]
116             h135[:oblig] = ( self[2] != FrazaAnot::TYP_SIE[3][1] )

```

```

117     end
118     h135
119 end# of to_arghash

121 def niedopasowania( arr137 )
122     # zwraca tablicę [[indeksy tych, które są w o a nie w 1], [indeksy tych,
    które są w 1 i nie w o ]]
123     arr0_137 = self
124     arr0_d = arr0_137.dup
125     arr1_d = arr137.dup
126     # odhaczamy kolejne tagi z jednej w drugiej i odwrotnie
127     arr0_137.each{ |elt137|
128         i137 = arr1_d.index( elt137 )
129         arr1_d.delete_at(i137) if i137}# delete at, a nie po prostu delete,
    żeby usunąć tylko jeden na raz
130     arr137.each{ |elt138|
131         i138 = arr0_d.index( elt138 )
132         arr0_d.delete_at(i138) if i138}
133     wy0_138, wy1_138 = [], []
134     dup2 = arr0_137.dup
135     arr0_d.each{ |elt139|
136         # znajdujemy ostatnie wystąpienie tagu niesymetrycznego
137         # i dodajemy id tej elipsy do tablicy, i odhaczamy ten tag w tablicy
    tagów,
138         # żeby go nie znaleźć po raz drugi.
139         i139 = dup2.rindex( elt139 )
140         wy0_138 « i139
141         dup2[i139] = nil}
142     dup2 = arr137.dup
143     arr1_d.each{ |elt140|
144         # znajdujemy ostatnie wystąpienie tagu niesymetrycznego
145         # i dodajemy id tej elipsy do tablicy, i odhaczamy ten tag w tablicy
    tagów,
146         # żeby go nie znaleźć po raz drugi.
147         i140 = dup2.rindex( elt140 )
148         wy1_138 « i140
149         dup2[i140] = nil}
150     return [wy0_138, wy1_138]
151 end# of niedopasowania.

155 def to_xml_attr_alpha # używana w eksporcie danych do XML-a.
156     # zwrócimy hasz, który Builder zapisze jako ciąg atrybutów w kolejności
    alfabetycznej.
157     # zakładamy, że jesteśmy tablicą tablic dwuelementowych, której dx jest
    wektorem atrybutów, a dy wektorem wartości (jesteśmy wynikiem
    aHash.to_a)
158     s140 = self.collect{ |kv140| kv140.collect{ |x140| x140.to_s }
    }.sort
159     s0_140 = s140.delete_at(0)
160     tail140=s140.collect{ |kv141| "#{kv141[0]}=#{kv141[1]}"

```

```

    }.join('" "') # zwróćcie uwagę, że zamykający cudzysłów dołącza
    join – ostatnie pozostanie nie domknięte, i o to właśnie chodzi.

162     return { s0_140[0].intern => [s0_140[1], if tail140.length>0 :
    tail140 end ].compact.join('" ').intern # ten intern sprawia, że
    cudzysłowy (ani nic innego) nie zostaną ucieknione.
163   }
164   end

168   end# of class Array.

171   class Hash

173     def to_argarr( trim=nil )
174       arg = self
175       if trim : tag0= "
176     else
177       if arg[:oblig] : tag0='fw:'
178     else tag0 = 'fl:'
179     end
180     end

182     if arg[:syncat]
183       typ=FrazaTyp.find( arg[:syncat] ).typ_symbol.downcase
184     else
185       # dwie ramki nie mają pola syncat, i nie wiem dlaczego,
186       # a nie mam czasu dążyć (trochę podrażyłem – plik test.rb)
187       if arg[:prep] : typ = 'prepn'
188     else typ = 'np'
189     end
190     end

192     tag0 += typ
193     tag0 += ':' unless typ =~ /(advp|sie)/

195     case typ
197     when 'advp'

199     when /^(adjp|np|nump)$/
200       tag0 += arg[:case]

202     when /^(prepadjp|prepn|prepnum)$/
203       tag0 += arg[:prep] + ':' + arg[:case]

205     when 'sie' # Ela chce wyróżniać „się” bezosobowe jako frazę luźną.
206       # tag0 += 'sie'
207     else
208       if typ==nil : tag0=':???'
209     else raise "nie rozpoznany typ frazy: #{typ}"
210     end
211     end # of case

213     # tag0+= ':'
214     [ tag0,
215       if arg[:sens]

```

```

216         arg[:sens].collect{|s| Sensy.ozn2opis( s )}.join(" ")
217         # 2008/07/18 Ela chce sensy pełnym słowem.
218     else ""
219     end ]
220 end # of to_argarr

223 def to_xml_attr_alpha
224     # zwrócimy się w postaci Builderowego ciągu atrybutów posortowanych
    alfabetycznie.
225     self.to_a.to_xml_attr_alpha
226 end

228 end # of class Hash

231 class String

233     def sens_pusty?
234         self == "" or self == Ramkowanie::UNDEF or self == nil
235     end

237     def to_arghash( ohash={} )
238         # to może być :zostawtyp => true lub :oblig => true/false
239         # przetwarza tag frazy (argumentu ramki) w hasz do zapisania w
        ramka.rbo
240         tag=self
242         zmientyp = (not ohash[:zostawtyp] )

244         ts=tag.split(':')
245         h=Hash.new
246         if ts[0] =~ /\^[a-z)]$/ # jeśli pasuje do fw lub fl, to oblig = true <->
        fw
247             h[:oblig] = ($1 == 'w')
248             ts.delete_at( 0 )
249         else
250             h[:oblig] =true
251         end

253         h[:oblig] = ohash[:oblig] if ohash.has_key?( :oblig )

255         case ts[0]

257         when 'advp$'

259         when /\^(adjp|np|nump)$/
260             h[:case] = ts[1]
261             ts[0]='np' if zmientyp

263         when /\^(prepadjp|prepn|prepnump)$/
264             h[:prep], h[:case] = ts[1], ts[2]
265             ts[0]='prepn' if ts[0]=='prepnump' and zmientyp

267         when 'sie' # Ela chce wyróżniać „się” bezosobowe jako frazę luźną.
268             # tago += 'sie'
269         end# of case

271         h[:syncat] = FrazaTyp.downhash[ ts[0] ]

```

```

273     # sprawdzamy, czy tag (z zapewne zamienionym typem) jest na liście
      elips
274     if zmientyp
275 # raise "tag niepoprawny!!!" unless Elipsa.lista_fraz.include?( ts.join('.') )
276     end
277     h
278 end# of to_arghash

281 def trimtyparg
282     self =~ /^(f[a-z]:)?(.*)$/
283     $2.to_s
284 end

287 def to_bool
288     if (sds = self.downcase.strip) == 'true' or sds == 't' or
289         self.to_i != 0 : true
290     else false
291     end
292 end # of to_bool

293 @@depolon = {
294     'ą' => 'a', 'ć' => 'c', 'ę' => 'e', 'ł' => 'l', 'ń' => 'n', 'ó' =>
295     'o', 'ś' => 's', 'ż' => 'z', 'ź' => 'z',
296     'Ą' => 'A', 'Ć' => 'C', 'Ę' => 'E', 'Ł' => 'L', 'Ń' => 'N', 'Ó' =>
297     'O', 'Ś' => 'S', 'Ż' => 'Z', 'Ź' => 'Z'
298 }

299 def odpolszcz
300     self.gsub(/([ąćęłńóśż])/) {|match192| @@depolon[ match192 ] }
301 end

303 def nkjp_escape
304     # 2009/9/10 po konsultacji z MW co będzie a co nie będzie poprawnym
      XML-em, co zgadza się z empirią niewyeskejpowanego cudzysłowu,
      eskejpuję trzy z czterech HTML-owych.
305     self.gsub(/&/, "&amp;"). ## gsub(/\\"/, "&quot;").
306     gsub(>/, "&gt;").gsub(</, "&lt;")
307 end

311 end # of class String

314 class ActiveRecord::Base
315     def self.pragmas_for_update
316         logger "=== pragmas for update #{Time.now.to_s(:db)}"
317         self.connection.execute " pragma locking_mode = exclusive; "
318         self.connection.execute " pragma synchronous=off; "
319     end

321     def self.pragmas_normal
322         self.connection.execute " pragma locking_mode = normal; "
323         self.connection.execute " pragma synchronous=normal; "
324     end

326     def self.max_created_at( conditions )

```



```
327     self.maximum( :created_at, :conditions => conditions ) ||
      '1969-06-29'.to_time
328   end
329
330   def self.unnull_akat_bliźniaki
331     self.connection.execute "update akapit_transzy set blizniaczy_id
      =(select akapit_transzy_id from akapit_transzy at1where
      at1.akapit_id= akapit_transzy.akapit_idand at1.akapit_transzy_id
      <> akapit_transzy.akapit_transzy_id)where blizniaczy_id is null"
332     # poniższe doprowadziło do tego, że przy kolejnej dolewce 30 akatów nie
      miało bliźniaka.
333     ## and akapit_id between # akapit_od and akapit_do
334   end
335
336 end # of ActiveRecord::Base
337
338 class Range
339
340   def min
341     if self.begin <= self.end
342       self.begin
343     end
344   end
345
346   def max
347     if self.begin <= self.end
348       self.end
349     end
350   end
351
352   def &( drugie )
353     # oczywiście nie możemy skonwertować do Array: zakresy mogą być
      gigantyczne.
354     if self.include?( drugie.min ) or self.include?( drugie.max ) or
355        drugie.include?( self.min ) or drugie.include?( self.max )
356       krańce = [self.max, self.min, drugie.min, drugie.max].sort
357       (krańce[1])..(krańce[2])
358     else
359       nil
360     end
361   end
362 end # of &
363
364 end # of class Range
365
366 class Zmienna4RadioButton < Object
367   attr_accessor :wartość
368 end
369
370 module XpointerCreator
371   # jego metodę używam w NowaSegmentacja i SgVariant.
372
373   def create_xpointers( orths_list, tokens_list, is_list=nil )
374     # tworzymy tablicę xpointerów odpowiadających orthom wedle
      xpointerów podanych tokenów. Nie sprawdzamy, czy dwie listy wejściowe
      są kompatybilne: w NowaSegmentacja zrobiliśmy to wcześniej, w
```

```

SgVariant zakładamy, że wariant dodany został dodany poprawnie (z
dokładnością do xpointerów).
378 # dii jest tablicą tokenów. Bierzemy ich xpointery, które mówią, od
którego znaku się token zaczyna i na którym kończy, i przeliczamy
xpointery dla dodawanych tokenów.

380 if is_list
381   orths_is_list = is_list
382 else
383   orths_is_list = tokens_list.collect { |t144| t144.orth }
384 end

386 ##      puts tokens_list.collect { |t144| t144.id }.join( ' ' )

389 md143 = nil

391 xps144 = tokens_list.collect { |t144|
392   logger.info "### t.xpointer of #{t144.id}:
#{t144.xpointer.inspect}"
393   md144 = /^(.*) ,(\d+),(\d+)\)$/ .match( t144.xpointer )
394   logger.info "###
#{md144.inspect}"

396   [ md144[ 1 ] + ', ', # dodajemy przecinek, bo xpointery mają postać
....ab,nr,nr".
397     md144[ 2 ].to_i, md144[3].to_i ]
398 }
399 # xps jest teraz tablicą trójek, z których każda zawiera początek xpointera,
czyli |string-range(txt_19.1-ab|, nr znaku od, liczbę znaków.

401 süßtaffel = []
402 orths_is_list.each_index { |i144|
403   # teraz tablicę orthów seg_is obrabiamy w ten sposób, że każdy orth
dzielimy na epsilon, czyli zamieniamy w tablicę pojedynczych znaków
404   orths_is_list[i144].split("").
405   # następnie przebiegamy indeksy tej tablicy i dla każdego z nich
wpisujemy do süßtaffel wartość odpowiedniego xpointera „od znaku”.
406   each_index { |j144|
407     süßtaffel « ( xps144[i144][1] + j144 )
408   }
409 }
410 # Tym sposobem süßtaffel zawiera teraz numery znaków,
odpowiadające kolejnym znakom orthów seg_is.
411 # Zauważmy, że nie muszą to być liczby ściśle kolejne: zdarzają się
przeskoki tam, gdzie w tekście jest spacja. Są to jednak z pewnością
liczby parami różne, a nawet – uporządkowane ściśle rosnąco.

413 sxps144 = [] # „Should-be XPointerS”
414 orths_list.each_index { |i145|
415   # Teraz obrabiamy orthy seg_should_be:
416   # pobieramy z süßtaffel do untertaffel tyle znaków, ile długość
kolejnego ortha...
417   ojl = orths_list[ i145 ].jlength

```

```

418     undertaffel = süßtaffel[ 0, ojl]
419     # i usuwamy je z süßtaffel:
420     süßtaffel = süßtaffel[ ojl, [süßtaffel.size - ojl, 1].max ]
422     # a następnie dopisujemy do tablicy nowych (szudbicznych)
    xpointerów nowy xpointer
424     sxps144 « ( xps144[ [xps144.size-1, i145].min ][0] + # to jest
    początek i-tego lub przedostatniego xpointera oryginalnego
425         "#{undertaffel[0]},#{undertaffel[-1]-
    undertaffel[0]+1})" # a to wartość „od znaku”, liczba
    znaków (+1 dodane 2010/2/24 po zgłoszeniu błędu przez
    analizujących wyjście). Domykamy nawias xpointera.
426     )
427 }
428 sxps144 # zwracamy „Should-be XPointerS”
430 end # of create_xpointers
433 end # of module XpointerCreator
436 ## # Local Variables:
437 ## # mode: ruby
438 ## # End:

```

## rola.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: rola
5  #
6  # opis :text
7  # rola_id :integer primary key
8  # opis_kr :string
9  # created_at :timestamp
10 # updated_at :timestamp
11 #
35 load 'interpretacja_anot.rb'
36 load 'sens_anot.rb'
37 # bo krzyczało "Expected interpretacja_anot to define InterpretacjaAnot"
39 class Rola < ActiveRecord::Base
41   has_many :uzytkownik
43   ROLE={
44     :zablokowany => {:rola_id => 0, :opis => "zablokowany", :opis_kr
    => "zablokowany"}.freeze,
45     :gosc =>          {:rola_id => 1, :opis => "gość (widzi
    uzgodnione)", :opis_kr => "gość"}.freeze,
46     :audytor =>       {:rola_id => 2, :opis => "audytor (widzi
    wszystko)", :opis_kr => "audytor"}.freeze,
47     :anotator =>      {:rola_id => 4, :opis => "anotator", :opis_kr
    => "anotator"}.freeze,

```

```

48      :zarzadca =>          { :rola_id => 8, :opis => "superanotator (jego
                             anotacja zawsze będzie zatwierdzona i ostateczna)", :opis_kr =>
                             "superanotator" }.freeze,
49      :admin =>             { :rola_id => 16, :opis => "administrator
                             (informatyczny)", :opis_kr => "admin" }.freeze
50    }.freeze

53    @@nazwy_ról = {}

55    ROLE.each_pair { |k206, v206|
56      @@nazwy_ról[ v206[:rola_id] ] = v206[:opis_kr]
57    }

60    def self.zrob_sie
61      unless self.find( :first )
62        ROLE.each_pair { |key128, val128|
63          self.connection.execute "insert into rola ( created_at,
                                   rola_id, opis, opis_kr) " +
64                                   " values( current_timestamp,
                                   #{val128[:rola_id]}, '#{val128[:opis]}',
                                   '#{val128[:opis_kr]}' )"
65          # pierwsze podejście, self.create( val ), powodowało, że zadane
                                   # przeze Mnie idy były ignorowane.
66        }
67      end
68    end # of self.zrob_sie.

70    zrob_sie unless self.find( :first )

72    def self.rid( rola )
73      ROLE[rola][:rola_id]
74    end

77    def self.usuwalny?( uzytkownik )
78      # Mówi, czy można użytkownika z taką rolą usunąć, czy tylko
                                   # zablokować.
79      # Nie usuniemy audytora, który pozostawił jakiś komentarz ani
                                   # anotatora, który coś zweryfikował,
80      # ani zarządcy, jeśli są akapity osądzone bądź osądzone.
81      if no_record?( :komentarz, uzytkownik ) and
82        ((not uzytkownik.ktoras_z_rol?( :anotator, :zarzadca,
                                   :zablokowany)) or
83         (no_record?( :akapit_transzy, uzytkownik ) and
84          no_record?( :statusy, uzytkownik ) and
85          no_record?( :interpretacja_anot, uzytkownik ) and
86          no_record?( :sens_anot, uzytkownik ))) and
87        ((not uzytkownik.ktoras_z_rol?( :zarzadca, :zablokowany)) or
88         AkapitTranszy.find_by_sql(
89           ["select * from akapit_transzy where
90            status=? or status=?",
91            AkapitTranszy::STATI[:osadzony],
92            AkapitTranszy::STATI[:osadzany]]).size
93           == 0)

```

```
92         return true
93     else return false
94     end
95 end # of self.usuwalny?

98 def self.admin?(rola_id)
99     self.rola?(rola_id, :admin)
100 end

103 def self.zarzadca?( rola_id )
104     self.rola?(rola_id, :zarzadca)
105 end

108 def self.anotator?( rola_id )
109     self.rola?( rola_id, :anotator )
110 end

113 def self.audytork?(rola_id)
114     self.rola?(rola_id, :audytork)
115 end

118 def self.wszewid?( rola_id )
119     # audytork lub zarzadca: ma prawo oglądać wszystko i dodawać
    komentarze
120     return ( self.rola?(rola_id, :audytork) or self.rola?(rola_id,
        :zarzadca) )
121 end

123 def self.gosc?(rola_id)
124     return self.rola?( rola_id, :gosc )
125 end

128 def self.rola?(rola_id, rola_symb)
129     # czy dany identyfikator roli odpowiada danemu symbolowi roli
130     if rola_id & self.rid(rola_symb) == 0
131         return false
132     else
133         return true
134     end
135 end

138 def self.ktoras_z_rol?( uzytkownik, * lista_rol )
139     # chcemy wołać tę metodę dla ( uzytkownik, [:rola1, :rola2, ...] )
140     # drugi argument może też być pojedynczym symbolem roli.
141     # gdzie uzytkownik może być Uzytkownik-iem lub idem Uzytkownika.

143     uzytkownik=Uzytkownik.find(uzytkownik) if uzytkownik.kind_of?(
        Integer )
144     lista_rol = [lista_rol] unless lista_rol.kind_of?( Array )

146     r_c = lista_rol.collect{ |r128|
147         if Rola.rola?(uzytkownik.rola_id, r128) : 1
148         else nil
149         end
    }.compact
150     return( r_c.size > 0 )
```

```

151     end # of self.ktoras_z_rol?.

154     def self.nazwa_rol( rola_id )
155       @@nazwy_rol[ rola_id ]
156     end

159     private

161     def self.no_record?(tab_symb, uzytkownik)
162       uzytkownik.no_record?( tab_symb )
163     end

168   end # of class

171   ## # Local Variables:
172   ## # mode: ruby
173   ## # End:

```

### **segmentation\_rozbieznosc.rb**

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: segmentation_rozbieznosc
5   #
6   # segmentation_rozbieznosc_id :integer primary key
7   # sg_choice_id :integer not null
8   # sg_variant_id :integer not null
9   # created_at :timestamp
10  # updated_at :timestamp
11  #

35  class SegmentationRozbieznosc < ActiveRecord::Base
36    has_one :punkt_protokolu, :as => :elt_protokol
37    belongs_to :sg_variant
38    belongs_to :sg_choice
39    has_many :token, :through => :sg_variant
40  end

43  ## # Local Variables:
44  ## # mode: ruby
45  ## # End:

```

### **sens\_anot.rb**

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: sens_anot
5   #
6   # sens_anot_id :integer primary key
7   # uzytkownik_id :integer
8   # akapit_transzy_id :integer not null
9   # token_id :integer not null
10  # interpretacja_id :integer not null

```

```

11 # sensy_id :integer not null
12 # automatycznie :boolean
13 # created_at :timestamp
14 # updated_at :timestamp
15 #

39 class SensAnot < ActiveRecord::Base
40   belongs_to :uzytkownik
41   belongs_to :interpretacja
42   ##   belongs_to :sensy_leksemu, :counter_cache => true
44   # counter cache chce pluralizować, więc dodałem do inflektora regułę,
45   # że „sens” i „anot” są niepoliczalne
46   belongs_to :akapit_transzy
47   belongs_to :sensy

49   def self.znajdz( tokid, param )
50     if param.kind_of?( Integer )
51       phash = { :akapit_transzy_id => param }
52     elsif param.kind_of?( Hash )
53       phash = param.dup
54     end
55     phash[:token_id] = tokid

57     self.find( :first,
58               :conditions => phash)
59   end

61 end

65 ## # Local Variables:
66 ## # mode: ruby
67 ## # End:

```

## sensy.rb

```

1 # == Schema Information
2 # Schema version: 51
3 #
4 # Table name: sensy
5 #
6 # sensy_id :integer primary key
7 # cz_m_id :integer not null
8 # cz_m_leksem_id :integer not null
9 # xmlid :text
10 # n :integer
11 # short_def :text
12 # long_def_xml :text
13 # long_def_html :text
14 # created_at :timestamp
15 # updated_at :timestamp
16 #

40 class Sensy < ActiveRecord::Base
41   belongs_to :cz_m

```

```

42 belongs_to :cz_m_leksem
43 has_many :interpretacja
44 has_many :sens_anot

47 @short_description = nil
48 @long_description = nil

50 def short_description
51   if ( num199 = self.n ) == 666666
52     lemat199 = "inne znaczenie"
53     num199 = "
54   else
55     lemat199 = self.cz_m_leksem.lemat
56     num199 = "#{num199}:"
57   end
58   @short_description ||=
59     "<span class='\"sc\"'>#{lemat199}</span>
        #{num199}&#8194;#{self.short_def}" # to jest półfret. Ćwierćfret:
        &#8197;

60 end

63 @@rends = Hash.new { |h193, key193|
64   raise "Sensy.@@rends: unrecognised key #{key193}" }
65 @@rends[ 'bold' ] = 'b'

68 def short_def_html
69   # w tej wersji słownika (2009/8/20) niepotrzebne: w shortdefach nie ma
        żadnych znaczników xml.
70   @rends_short = []
71   rend_no = 0
72   ssd193 = self.short_def.gsub( /<hi rend="(.*?)">/ # dopasowanie
        niezachłanne (minimalne)
73                                     ) { |match193|
74     r193 = ( @rends_short[rend_no] = @@rends[ $1 ] )
75     rend_no += 1
76     "<#{r193}>" }
77   rend_no = -1
78   ssd193.gsub( /<\/hi>/ ) { |match193|
79     rend_no += 1
80     "<\/#{@rends[rend_no]}>"
81   }
82 end

85 def long_description
86   @long_description ||= self.long_def_html
87   # prawdopodobnie będziemy podmieniać znaczniki XML-owe
        sense_inventory na HTML-owe.
88 end

90 def long2html
91   # gromadkę podsensów ujmujemy w listę numerowaną
92   sld193 = "<ol>\n" + self.long_def_xml + "\n</ol>"
93   # podsens w element listy

```



```

94     sld193 = sld193.
95       gsub(/<sense n="(.*?)">/) { |match192| "<li value=\"#{s1}\">>".
96       gsub( /<\sense>/, "" )

98     @rends_long = []
99     rend_no = 0
100    sld193 = sld193.gsub( /<hi rend="(.*?)">/ # dopasowanie
    niezachłanne (minimalne)
101      ) { |match193|
102        r193 = ( @rends_long[rend_no] = @@rends[ s1 ] )
103        rend_no += 1
104        "<#{r193}>" }
105    rend_no = -1
106    sld193 = sld193.gsub( /<\hi>/ ) { |match193|
107      rend_no += 1
108      "</#{@rends_long[rend_no]}>"
109    }

111    sld193 = sld193.gsub( /<\/* def>/, "" ). # elementy def zwykłą
    czcionką
112    gsub( /<\/* cit( type="dicteg")* >/, "" ). # cit-y ignorujemy,
    a dokładniej: obsługujemy poniżej
113    gsub(/<quote>/, " &#8194;•<i>&nbsp;" ).
114    gsub(/<\quote>/, "</i> ").
115    gsub(/<b([awuiozAWUIOZ])( +)/, '\1&nbsp;') # wiszące litery

117    self.long_def_html = sld193
118    self.save!
119  end # of long2html

122  def self.long2html( * query_params )
123    sf193 = self.find( * query_params )
124    if sf193
125      if sf193.kind_of?( Sensy )
126        sf193.long2html
127      else
128        sf193.each { |sens193|
129          @@currid = sens193.id
130          sens193.long2html }
131      end
132      return true
133    else return false
134    end
135  end # of self.long2html.

137  # treść opisów do „inne”
138  INNE_short = "
139  INNE_long = "Znaczenie inne od podanych powyżej. Niniejsze " +
140    "„znaczenie” powinno być wybierane tylko w wyjątkowych i dobrze " +
141    "uzasadnionych wypadkach."
142  INNE_long_html = "<p>" + INNE_long + "</p>"

144  def self.dopisz_inne

```

```

145         # każdy cz_m_leksem ma mieć sens <inne>
146         CzMLeksem.dopisz_inne
147     end
151 end

```

### **sensy\_leksemu.rb**

```

23 # == Schema Information
24 # Schema version: 1
25 #
26 # Table name: sensy_leksemu
27 #
28 # sensy_leksemu_id :integer primary key
29 # sensy_id :integer
30 # leksem_id :integer
31 # nowy :boolean
32 # sens_anot_count :integer default(0)
33 # sens_zweryf_count :integer default(0)
34 #
36 class SensyLeksemu < ActiveRecord::Base
37   belongs_to :sensy
38   belongs_to :leksem
39   has_many :sens_anot
41   def usun_smiec
42     if self.nowy?
43       SensyLeksemu.delete_all("nowy = 't' and sens_anot_count <= 0")
44     end
45   end
47 end

```

### **sensy\_sensemu.rb**

```

23 # == Schema Information
24 # Schema version: 1
25 #
26 # Table name: sensy_sensemu
27 #
28 # lemat :text
29 # klasa_sem :text
30 # sensy_id :integer
31 # sens_ozn :text
32 # sens_opis :text
33 # nowy :
34 # sens_anot_count :
35 # sens_zweryf_count :
36 #
38 class SensySensemu < ActiveRecord::Base
40   def nowy?

```

```
41     if self.nowy == 't' : return true
42     else return false
43     end
44 end
46 end
```

### **sentences\_rozbieznosc.rb**

```
1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: sentences_rozbieznosc
5  #
6  # sentences_rozbieznosc_id :integer primary key
7  # token_id :integer not null
8  # czy_konczy_zdanie :boolean not null
9  # created_at :timestamp
10 # updated_at :timestamp
11 #

35 class SentencesRozbieznosc < ActiveRecord::Base
36   has_one :punkt_protokolu, :as => :elt_protokol
37   belongs_to :token
38 end
```

### **sg\_choice.rb**

```
1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: sg_choice
5  #
6  # sg_choice_id :integer primary key
7  # akapit_id :integer not null
8  # dodany :boolean
9  # xmlid :text
10 # created_at :timestamp
11 # updated_at :timestamp
12 #

36 class SgChoice < ActiveRecord::Base
37   belongs_to :akapit
38   has_many :sg_variant, :dependent => :destroy # klauzula destrukcji
   dodana 2009/7/4, dla zniszczenia niefortunnego „niedostrzeganiem”
   z akapitu 19 (migracja 010)

40   def token # to, że nie jest to has_many, ma poważną przyczynę.
41     Token.find( :all, :conditions => { :sg_choice_id => self.id },
42               :order => :kolejnosc )
43   end# of token

46   def last_chosen_token
47     unless defined?( @last_chosen_token ) and @last_chosen_token
```

```

48         @last_chosen_token = Token.find( :first,
49                                           :conditions => { :sg_choice_id =>
50                                                         self.id, :chosen => true },
51                                           :order => "kolejnosc desc" )
52     end
53   end
54
55   def segmentation_xmlid
56     # używana do oznaczenia zdania w wypluwce.
57     # Zdanie chcemy oznaczyć ostatnim tokenem.
58     if t213 = last_chosen_token
59       t213.segmentation_xmlid
60     end
61   end
62
63   def morphosyntactic_xmlid
64     # używana do oznaczenia zdania w wypluwce.
65     # Zdanie chcemy oznaczyć ostatnim tokenem.
66     if t213 = last_chosen_token
67       t213.morphosyntactic_xmlid
68     end
69   end
70 end
71
72 @tokids = nil
73
74 def tokids
75   unless @tokids
76     @tokids = self.token.collect{ |t128| t128.id }
77   end
78   @tokids
79 end
80 end
81
82 end

```

### sg\_variant.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: sg_variant
5  #
6  # sg_variant_id :integer primary key
7  # sg_choice_id :integer not null
8  # xmlid :text
9  # dodany :boolean
10 # created_at :timestamp
11 # updated_at :timestamp
12 #
13
36 require 'ramkowanie'
37
39 class SgVariant < ActiveRecord::Base
40   belongs_to :sg_choice

```

```

41     has_many :token, :order => :kolejnosc
42     has_many :sg_variant_anot

44     include XpointerCreator

46     @tokids = nil

48     def tokids
49       unless @tokids
50         @tokids = self.token.collect { |t| t.id }
51       end
52       @tokids
53     end

56     def równoległy_oryginalny
57       unless self.token[0].dodany?
58         # fajnie byłoby przechowywać informację o pochodzeniu tokenów w
          naszej kolumnie, ale ciężko znaleźć miejsce/a, w których taką
          informację byłoby wygodnie wpisać: w momencie tworzenia nowego
          wariantu zwykle jeszcze nie ma on tokenów.
59         self
60       else
61         self.class.find( :first, :conditions =>
62           " sg_choice_id=#{self.sg_choice_id} and " +
63           " ( select dodany from token where " +
64           " token.sg_variant_id=sg_variant.sg_variant_id
65           )='f' "
66         )
67       end # of równoległy_oryginalny

70     def popraw_xpointers
71       if self.token[0].dodany? and self.token[0].chosen?
72         # poprawiać jest sens jedynie xpointery wariantów dodanych.
73         # ograniczam też do wariantów z tokenami wybranymi, bo tylko one
          się liczą dla wypłuwki.
74         orths_list = self.token.collect { |t223| t223.orth }
75         puts self.token.collect { |t223| t223.id }.join( ' ' )
76         tokens_list = self.równoległy_oryginalny.token
77         xpo = create_xpointers( orths_list, tokens_list )
78         sto = self.token
79         xpo.each_index { |i223|
80           if ( old_xpo = sto[ i223 ].xpointer ) != xpo[ i223 ]
81             sto[ i223 ].xpointer = xpo[ i223 ]
82             sto[ i223 ].save!
83             puts "#{sto[i223].id} było #{old_xpo} jest #{xpo[i223]}"
84           end
85         }
86       end
87     end # of popraw_xpointers

90     def self.popraw_xpointers( ids )
91       [ self.find( ids ) ].flatten.each { |sgv223|

```

```

92     sg_v223.popraw_xpointers
93   }
94   end

97   def self.tokododane
98     self.find( :all, :conditions =>
99       " exists (select * from token where " +
100       " sg_variant_id=sg_variant.sg_variant_id and " +
101       " dodany='t' and chosen='t' )"
102     )
103   end # of self.tokododane

106   def self.sprawdz_xpointer( id, oryginalny=nil )
107     if oryginalny
108       sg_v = self.find( id ).rownolegly_oryginalny
109     else
110       sg_v = self.find( id )
111     end
112     Token.sprawdz_xpointer( sg_v.token[0].id )
113   end

115   end # of class

sg_variant_anot.rb

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: sg_variant_anot
5   #
6   # sg_variant_anot_id :integer primary key
7   # sg_choice_id :integer not null
8   # sg_variant_id :integer not null
9   # chosen :boolean not null
10  # akapit_transzy_id :integer not null
11  # uzytkownik_id :integer
12  # created_at :timestamp
13  # updated_at :timestamp
14  #

38  class SgVariantAnot < ActiveRecord::Base

40    belongs_to :akapit_transzy
41    belongs_to :sg_choice
42    belongs_to :sg_variant

44    def token
45      self.sg_variant.token
46    end

48    ##   def self.znajdz( tokid, akat_id )
50    ##     self.find( :first, :conditions => { :token_id => tokid,
      :akapit_transzy_id => akat_id })
52    ##   end

```

```

55     def bliźniaczy
56       @bli_hasz = {
57         :sg_variant_id => self.sg_variant_id,
58         :akapit_transzy_id => self.akapit_transzy.blizniaczy_id
59       }
60       self.class.find( :first, :conditions => @bli_hasz )
61     end # of bliźniaczy

62     def skopiuj_na_bliźniaczy
63       bli = self.bliźniaczy # ta met. tworzy @bli_hasz
64       bli ||= self.class.new( @bli_hasz )
65       bli.sg_choice_id = self.sg_choice_id
66       bli.chosen = self.chosen
67       bli.uzytownik_id= self.uzytownik_id
68       bli.save!
69     end # of skopiuj_na_bliźniaczy

70   end

71   ## # Local Variables:
72   ## # mode: ruby
73   ## # End:

```

## statusy.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: statusy
5  #
6  # statusy_id :integer primary key
7  # akapit_transzy_id :integer not null
8  # akapit_id :integer not null
9  # segmentation :integer default(-1)
10 # sentences :integer default(-1)
11 # morphosyntactic :integer default(-1)
12 # word_senses :integer default(-1)
13 # syntactic_words :integer default(-1)
14 # named_entities :integer default(-1)
15 # syntactic :integer default(-1)
16 # segmentation_uzid :integer
17 # sentences_uzid :integer
18 # morphosyntactic_uzid :integer
19 # word_senses_uzid :integer
20 # syntactic_words_uzid :integer
21 # named_entities_uzid :integer
22 # syntactic_uzid :integer
23 # segmentation_odrzucony :boolean
24 # sentences_odrzucony :boolean
25 # morphosyntactic_odrzucony :boolean
26 # word_senses_odrzucony :boolean
27 # syntactic_words_odrzucony :boolean

```

```

28 # named_entities_odrzucony :boolean
29 # syntactic_odrzucony :boolean
30 # created_at :timestamp
31 # updated_at :timestamp
32 # transza_id :integer
33 #

57 class Statusy < ActiveRecord::Base

59   belongs_to :akapit_transzy
60   belongs_to :akapit # używane m.in. przez o15_odzółć.rb
61   belongs_to :transza

64   STATI = AkapitTranszy::STATI
65   DZIAŁAJĄCE = PoziomyAnotacji::DZIAŁAJĄCE

68   def set_poziom( poziom, wartosc, bez_uS=nil )
69     logger.info "### (statusy id #{self.id}).set_poziom: poziom
        #{poziom.inspect}, wartosc #{wartosc.inspect}, bez_uS
        #{bez_uS.inspect}"
70     if wartosc.kind_of?( Symbol )
71       wert = STATI[ wartosc ]
72     else wert = wartosc
73     end
74     self.send( "#{poziom}=", wert )
75     self.save!
76     self.propaguj_dopuszczalność( poziom ) # ta metoda mnie zapisze.
77     self.akapit_transzy.update_Status unless bez_uS
78   end # of set_poziom

81   def propaguj_dopuszczalność( poziom )
82     if self.reload.send( poziom ) >= STATI[ :zweryfikowany ]
83       PoziomyAnotacji.nastepniki( poziom ).each { |poz197|
84         if self.send( poz197 ) == STATI[ :nie_dopuszczony ]
85           self.set_poziom( poz197, STATI[ :dopuszczony ] )
86         end
87       }
88     else
89       PoziomyAnotacji.wyzsze( poziom ).each { |poz197|
90         self.set_poziom( poz197, STATI[ :nie_dopuszczony ] )
91       }
92     end
93     self.save!
94   end # of propaguj_dopuszczalność

97   def uzid( poziom )
98     self.send( (poziom.to_s + '_uzid') )
99   end # of uzid

102   def set_uzid( poziom, uzid )
103     self.send( (poziom.to_s + '_uzid='), uzid )
104   end # of set_uzid

107   def statuski

```



```

108     PoziomyAnotacji.poziomy_x.collect{ |poz|
109       self.send( poz ) }
110   end # of statuski

113   def self.odźóć( poziomy = DZIAŁAJĄCE )
114     [poziomy].flatten. # jak widzicie, można też podać tylko jeden
      poziom
115     each { |poz|
116       self.find( :all, :conditions => "
        #{poz}={STATI[:zweryfikowany]} and " +
117         "statusy_id = (select min( statusy_id ) from statusy sx
          where sx.akapit_id = statusy.akapit_id) " ).
118     each { |st|
119       st.akapit.token_sup.each { |tsu|
120         tsu.odźóć( poz )
121         puts tsu.id
122       }
123     }
124   }
125   end # of self.odźóć

128   private

131   def self.autozatwierdź_segmentation ## ( akapit_od, akapit_do )
132     # użyta niejawnie w wierszu 676
133     if self.find_by_sql( [" select count(*) as ilefromstatusy s inner
134       join akapit_transzy akat using( akapit_transzy_id, akapit_id
        )where      akapit_id in (select distinct akapit_id from token
          where sg_choice_id is null)and      s.segmentation < ?",
135         STATI[:zweryfikowany]]
136       )[ 0 ].ile.to_i != 0
137       return Akapit.autozatwierdź_segmentation ## ( akapit_od,
        akapit_do )
138     end
139   end # of self.autozatwierdź_segmentation.

143   def self.dopusć_minimalne ## ( akapit_od, akapit_do )
144     PoziomyAnotacji.minimalne.each{ |poz|
145       if self.find( :first, :conditions => { poz => -1})
146         self.connection.execute " update statusy set #{poz}=0 where
          #{poz}=-1" ## +
147         ## " and akapit_id between #{akapit_od} and #{akapit_do}"
148       end
149       self.send( "autozatwierdź_#{poz}" ) ## , akapit_od, akapit_do
150     }
151     true
152   end # of self.dopusć_minimalne

161   def self.uzupełnij_tabelę ## ( akapit_od, akapit_do )
162     id_null = " select akapit_transzy_id, at.akapit_id,
163       at.transza_id from akapit_transzy at left outer join statusy " +

```

```

164         " using( akapit_transzy_id )" +
165         " where statusy_id is null "
166     if self.find_by_sql " select 1 where exists (#{id_null })"
167         raise "Table akapit_transzy is empty!" unless
168         AkapitTranszy.find( :first )
169         self.connection.execute( "insert into statusy( created_at,
170                                akapit_transzy_id, akapit_id, transza_id ) " +
171                                id_null.sub( "select", "select
172                                current_timestamp, " ) )
173         self.dopusć_minimalne ## ( akapit_od, akapit_do )
174         true
175     else
176         nil
177     end # of main unless,
178 end # of self.uzupełnij_tabelę.

179 def self.uzupełnij_transza_id
180     ret204=false
181     if ct203 = self.count( :conditions => "transza_id is null" ) != 0
182         puts "#{ct203} statusów beztranszowych – utranszawiam"
183         self.connection.execute " update statusy set transza_id=( " +
184                                "select transza_id from akapit_transzy at " +
185                                " where at.akapit_transzy_id=statusy.akapit_transzy_id) where "
186                                +
187                                "transza_id is null"
188         return true
189     else
190         return false
191     end
192 end # of self.uzupełnij_transza_id

193 ## Transza.zrob_transze # żeby zainicjować tę klasę, co spowoduje
194 ewentualne dotworzenie akapitów transzy.
195 ## uzupełnij_tabelę
196 # wykonałoby w szczególności self.dopusć_minimalne, które z kolei
197 zatwierdzą segmentacje, które można automatycznie.

201 def self.dopusć_word_senses
202     if DZIAŁAJĄCE.include?( :word_senses )
203         coś_zrobiłam185 = false
204         self.find( :all, :conditions => [ "morphosyntactic>=? and
205                                word_senses<?",
206                                STATI[:zweryfikowany],
207                                STATI[:dopuszczony]] ).each {
208                                |stat185|
209                                stat185.propaguj_dopuszczalność( :morphosyntactic )
210                                coś_zrobiłam185 = true unless coś_zrobiłam185
211                            }
212         coś_zrobiłam185
213     end
214 end # of self.dopusć_word_senses

```

```

214 dopuść_word_senses

217 def self.autozatwierdź_word_senses
218   dopuść_word_senses # propaguj_dopuszczalność zawiera
                        # utozatwierdzenie WSD, więc poniższe jest obszarne i powinno zwrócić
                        # false.
219   # piszemy czysty żywy SQL, bo WSD są poziomem maksymalnym i nie
                        # ma co wobec tego propagować dopuszczalności.
220   max188 = Statusy.maximum( :updated_at )
221   puts Time.now.to_s( :db )
222   puts "max188= " + max188.to_s( :db )

224   self.connection.execute( "update statusy set word_senses=16, " +
225                             " word_senses_uzid=#{Uzytkownik.natror.id},
                        " +
226                             " updated_at=current_timestamp where " +
227                             " word_senses=0 and " + # czyli jesteśmy
                        ms-zweryf.
228                             " ( (not exists (select interpretacja_id
                        from interpretacja i " +
229                             " inner join token tok using( token_id ) " +
230                             " where i.cz_m_leksem_id is not null " +
231                             " and i.disamb='t' and
                        tok.akapit_id=statusy.akapit_id ))" +
232   # " or ( statusy.morphosyntactic>=16 and not exists " +
233   # " ( select interpretacja_id from interpretacja i " +
234   # " inner join token tok using( token_id )" +
235   # " where i.cz_m_leksem_id is not null " +
236   # " and i.disamb='t' and tok.akapit_id=statusy.akapit_id ) )" +
237   " )"
238   )
239   # był tu błędny człon koniunkcji where i.disamb='t' – gdy nie ma
                        # zawężenia statusu ms do zweryfikowanych. To spowodowało błędne
                        # oznaczenie 141 tokenów w 57 akapitach.
240   # teraz dla danego akapitu ma albo nie być wcale żadnego ms-tagu
                        # sensownego, albo, jeśli akapit jest ms-zweryf., wystarczy, że nie ma
                        # sensownej dyzambiguacji.

242   puts Time.now.to_s( :db )
243   if max188 < Statusy.maximum( :updated_at )
244     AkapitTranszy.find_by_sql( [ "select akapit_transzy.* from
                        akapit_transzy inner join statusy s using( akapit_transzy_id )
                        where s.updated_at>=?", max188 ] ).
245       each { |akat193|
246         akat193.update_Status
247         puts akat193.id
248       }
249     return true
250   else return false
251   end
252 end # of self.autozatwierdź_word_senses

```

```

255     def autozatwierdź_word_senses
256     if self.reload.morphosyntactic >= STATI[ :zweryfikowany ]
257         ## robimy dla # :word_senses
258         if self.class.find_by_sql(
259             # 2009/10/22 kolejna próba ustawienia
                # testu: poprzednia dała 600 akapitów
                # błędnie zatwierdzonych.
260             # 2009/10/26 kolejna próba dała 326
                # akapitów błędnie zatwierdzonych jako
                # pustospełnione.
261             "select count(*) as ile from
                interpretacja i inner join token tok
                using( token_id ) where
                i.cz_m_leksem_id is not null and
                i.disamb='t' and
                tok.akapit_id=#{self.akapit_id} "
262             )[0].ile.to_i == 0
263             ## not self.akapit.reload.primus_sensibilis
264             # to wydaje się nie działać i prowadzić do totalnej katastrofy.
265             logger.info "#### ak. #{self.akapit_id} " +
266                 "akat #{self.akapit_transzy_id} wsd zatw. autom.
                (pustospełnione)."
267             self.reload.set_poziom( :word_senses, STATI[ :zweryfikowany ] )
268             self.word_senses_uzid = Uzytkownik.natror.id
269             self.save!
270         else
271             logger.info "#### ak. #{self.akapit_id} akat
                #{self.akapit_transzy_id} ma sensy."
272         end
273     end # of if morphosyntactic
274
275     end # of instance autozatwierdź_word_senses
276
277 end
278
281 ## # Local Variables:
282 ## # mode: ruby
283 ## # End:

```

### tagset.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: tagset
5  #
6  # tagset_id :integer primary key
7  # typ :text
8  # lewe :text
9  # prawe :text
10 # created_at :timestamp
11 # updated_at :timestamp

```

```

12  #

36  # Tabela ta jest poniekąd polimorficzna: zawiera różne typy reguł (aliases,
    # counteralias, attr, counterattr, pos), a ich obiekty »prawe« są odpowiednich
    # (różnych) typów:

37  #
38  # dla »aliases« – regexp klas gramatycznych danej części mowy,
39  # dla »counteralias« – string część mowy danej klasy gramatycznej;
    # niejednoznaczne dla »ger«,
40  # dla »attr« – tablica wartości danego atrybutu,
41  # dla »counterattr« – string atrybut danej wartości, w Tagsecie NKJP 0.2
    # mocno niejednoznaczne
42  # dla »pos« tablica przysługujących atrybutów postaci [atryb] lub [atryb,
    # :optional].

44  require 'ramkowanie'

46  class Tagset < ActiveRecord::Base
47    serialize :prawe

50    def self.zainicjuj( force=nil )
51      if force or not self.find( :first )
52        ##          self.connection.execute "delete from tagset "
53        Tagset.delete_all
54        require 'ftools' # for File.copy etc.
55        tagsetname='gramdata/tagset.cfg'
56        ##          self.connection.execute("pragma synchronous=off;")
57        currtype = nil

61        nowytyp = Proc.new { |match|
62          currtype = match.downcase }

64        typ_dobry = Proc.new {
65          %w( aliases attr pos ).include?( currtype ) }

67        File.open(tagsetname) {|f|
68          f.each {|line|
69            line.chomp!.strip!.downcase! # the last added 2010/6/16 while
            fixing #356.

71            case line
72            when /\^[^(+)\]\$/
73              match = $1
74              nowytyp.call( match )
75              puts currtype

77            when /\^[a-z].* =/
78              if typ_dobry.call
79                # wpisujemy do tabeli w bazie danych
80                lewe, prawe = line.split( '=' ).collect{|x| x.strip }
81                case currtype
82                when 'aliases' # w wersji tagsetu z 2009/06/23 nie ma
                    aliasów i nie są one używane w Anotatorni.

```

```

83      # (są natomiast używane części mowy, przechowywane
      # w tabeli cz_m, powiązanej z tabelą klasa_gram i używanej na
      # poziomie WSD.
84      right = Regexp.new( '(' + Regexp.escape( prawe ) + ')' )
85      prawe.split('/').each{ |klasa|
86          self.create(
87              :typ => 'counteralias',
88              :lewe => klasa,
89              :prawe => lewe)
90      }

92      when 'attr'
93          prawes = prawe.split
94          right = prawes
95          prawes.each{ |attr|
96              self.create(
97                  :typ => 'counterattr',
98                  :lewe => attr,
99                  :prawe => lewe)
100          }

102      when 'pos'
103          right = prawe.to_s.split.collect{ |pos|
104              if pos =~ /\^[(.*)\]$/: [$1, :optional]
105              else [pos]
106              end
107          }
108      end

110      tags = self.create(
111          :typ => currtype,
112          :lewe => lewe,
113          :prawe => right
114      )
115      end # of when /a-z/
116      end # of case line
117  }}

119      # dopiszemy :cololate tym atrybutom, które mają wartości
      # dwukropkowane, 2009/02/11 jest to atrybut numer_impt,
      # a 2009/03/04 – numberperson_impt.
120      true
121      end # of main condition
122      end # of self.zainicjuj

124      zainicjuj

126      ATTR2PL = Hash.new { |h, klucz|
127          h[klucz] = [ klucz, :f ] }

129      ATTR2PL.update(
130          # w bierniku, bo „nie wygląda na poprawną •”
131          {

```

```

132      'accentability' => ['akcentowość', :f],
133      'accommodability' => ['akomodacyjność', :f],
134      'agglutination' => ['aglutynacyjność', :f],
135      'aspect' => ['aspekt', :m],
136      'aspect_aglt' => ['aspekt przyczepnika', :m],
137      'aspect_będzie' => ['aspekt »będzie«', :m],
138      'aspect_pact' => ['aspectum participii activi', :m],
139      'aspect_pant' => ['aspekt partycyphu anteriorycznego',
140      :m],
141      'aspect_pcon' => ['aspekt partycypla
kontemporaryjnego', :m],
142      'case' => ['przypadek', :m],
143      'case_depr' => ['przypadek deprecjatywu', :m],
144      'case_prep' => ['przypadek prepozycjonatu', :m],
145      'case_siebie' => ['przypadek »siebie«', :m],
146      'degree' => ['stopień', :m],
147      'gender' => ['rodzaj', :m],
148      'gender_depr' => ['rodzaj deprecjatywu', :m],
149      'gender_ger' => ['genum gerundii', :m],
150      'gender_numcol' => ['rodzaj numeratu
kolekcyjnego', :m],
151      'negation' => ['przeczenie', :n],
152      'number' => ['liczbę', :f],
153      'number_depr' => ['liczbę deprecjatywu', :f],
154      'numberperson_impt' => ['liczboosobę rozkaznika', :f],
155      'person' => ['osobę', :f],
156      'post-prepositionality' => ['poprzyimkowość', :f],
157      'vocalicity' => ['wokaliczność', :f]
158    }
159  )

160  @@colonates = self.find( :all, :conditions =>
161    "typ='attr' and prawe like '%:%' " ).collect
162    {|ts|
163      ts.lewe }
164
165  def self.colonates
166    @@colonates
167  end
168
169  def self.colonate?( attr )
170    @@colonates.include?( attr )
171  end
172
173  POPRAWNY = {
174    :m => 'poprawny',
175    :f => 'poprawną',
176    :n => 'poprawne'}
177
178  ATRYBUTU = Hash.new( 'atrybutów' )
179
180  ATRYBUTU.merge!( 1 => 'atrybutu' )
181
182  @@bez_igna = true

```

```

183 # W skrypcie sprawdzającym tagi dla ŁD zmienimy wartość tej zmiennej.
186 def self.znajdz( typ, lewe, like=nil )
187   ign_clause = Proc.new { |cond|
188     if typ == 'pos' and @@bez_igna
189       cond[ 0 ] += " and lewe<>'ign' "
190       # klauzula wykluczająca ign spośród rozważanych tagów.
191     end }
192   unless like
193     condits = [ " typ = ? ", typ ]
194     ign_clause.call( condits )
195     if lewe == :all
196       self.find( :all, :conditions => condits )
197     else
198       condits[0] += " and lewe = ? "
199       condits[2] = lewe
200       self.find(:first, :conditions => condits )
201     end
202   else # the lewe argument is like
203     condition = [ " typ = ? and lewe like ? escape '\\ ' ", typ, lewe ]
204     ign_clause.call( condition )
205     self.find( :all, :conditions => condition )
206   end # of unless like
207 end # of self.znajdz

209 definition=" def self.znajdz_xxx( lewe, like=nil )   self.znajdz(
'xxx', lewe, like ) end def znajdz_xxx( lewe, like=nil )
self.class.znajdz_xxx( lewe, like ) end"

211 # definiujemy metody klasy i egzemplarza znajdz_pos
212 eval definition.gsub( 'xxx', 'pos' )

214 # definiujemy metody klasy i egzemplarza znajdz_attr
215 eval definition.gsub( 'xxx', 'attr' )

217 # definiujemy metody klasy i egzemplarza znajdz_counterattr
218 eval definition.gsub( 'xxx', 'counterattr' )

220 def trash
221 end

223 @@preferred = Hash.new

225 self.znajdz_attr( '%\_preferred', :like ).each{ |atr|
226   alewe = atr.lewe
227   a_s = alewe.split('_')
228   @@preferred[[ a_s[0], a_s[1] ]] = alewe
229 }

231 def self.preferred?( atr, gc )
232   @@preferred[ [atr, gc] ]
233 end

235 def self.preferreds
236   @@preferred

```



```

237     end

239     def self.attr_rx( attr )
240       za = znajdz_attr( attr )
241       if za : Regexp.new( '^(' + za.prawe.collect{ |pr|
242         Regexp.escape( pr )}.join('/') + ')$', 'i' )
243       else nil
244       end
245     end

246     def attr_rx( attr )
247       self.attr_rx( attr )
248     end

250     @@lemmatic = {
251       'siebie' => Proc.new { |lemat|
252         if lemat=='siebie' : nil
253         else 'jedynym dopuszczalnym lematem dla k.g. siebie jest
254           »siebie«'
255         end }
256     }

257     def self.check_tag( tag, lemat )
258       # zwrócimy true lub [false, <komunikat co źle>, <tag z podkreśleniem na
259       czerwono>]
260       tag_split=tag.split(':')
261       # znaleźć układ tagu dla klasy gramatycznej

262       zwrotka = true
263       komunikat = nil
264       redtag = tag
265       rozpoznanie = []

266       kg_pos = znajdz_pos( tag_split[0] )

267       if kg_pos
268         kg = tag_split[0]
269         atrybuty = kg_pos.prawe
270         atryb0, atryb1= atrybuty.dx, atrybuty.dy

271         # zgodnie z radą MW sprawdzamy dla każdej pozycji układ_tagu, czy
272         odpowiednia pozycja tag_split daje się zinterpretować jako taki właśnie
273         atrybut.
274         unless komunikat
275           tag_split.delete_at(0)
276           errmsg = []
277           ile_opt = atryb1.compact.size
278           tag_split.each_index{ |i|
279             if self.colonate?(atryb0[i])
280               # (2009/02/11) korzystamy z tego, że atrybuty dwukropkowe,
281               mianowicie numberperson_impt, są wymagane.
282               tag_split[i, 2] = tag_split[ i, 2 ].join(':')
283             end
284             platr = ATTR2PL[atryb0[i]]

```

```

285     poprawny_atryb = "#{POPRAWNY[platr[1]]} #{platr[0]}"
286     niepoprawny = "poz. #{i+2} (»#{tag_split[i]}«) nie wygląda na
      #{poprawny_atryb}"
287     nadmiarowy = "atrybut #{i+2} (#{tag_split[i]}) jest
      nadmiarowy"
288     if atryb0[i]
289       if tag_split[i] =~ attr_rx( atryb0[i] )
290       else
291         # korzystam z założenia, że atrybuty opcjonalne są na końcu
292         if atryb1[i] == :optional # czyli mamy atrybut opcjonalny i
          człon tagu nie pasuje do niego. I dalej już są same opcjonalne.
          Korzystam też z tego, że nie ma więcej niż dwóch atrybutów
          opcjonalnych.
293         # zakładam też, że opcjonalne, jeśli występują, to w takim
          porządku jak w tagsecie.
294         dalej = true
295         err_opt = []
296         until (not atryb0[i]) or (not dalej)
297           attr_rx = znajdz_attr( atryb0[i] ).prawe
298           if tag_split[i] =~ attr_rx
299             dalej = false
300             err_opt = nil
301           else
302             err_opt « niepoprawny
303             atryb0.delete_at( i )
304             atryb1.delete_at( i )
305             ile_opt -= 1
306           end
307         end
308         errmsg += err_opt if err_opt

310       else # atrybut obowiązkowy i niepoprawny
311         errmsg « niepoprawny
312       end
313     end
314     else
315       errmsg « nadmiarowy
316     end
317   }

319   brak_atr = atryb0.size - tag_split.size - ile_opt
320   if brak_atr > 0
321     ile_opc = atryb1.compact.size
322     if ile_opc > 0 : brak_atryb = "od #{brak_atr-ile_opc} do
      #{brak_atr}" # jeśli są opcjonalne we wzorcu
323     else brak_atryb = brak_atr
324     end
325     errmsg « "brakuje #{brak_atryb} #{ATRYBUTU[brak_atr]}"
326   end

328   if pro = @@lemmatic[ kg ]

```

```

329         if lemma_msg = pro.call( lemat )
330             errmsg « lemma_msg
331         end
332     end

334     if errmsg[0]
335         komunikat = errmsg.join(", ")
336         zwrotka = false
337     end

339     end # of unless komunikat

341     return [zwrotka, komunikat].compact

343     else
344         return [false, 'Nie rozpoznana klasa gramatyczna']
345     end

347 end# of self.check_tag

350 def self.matches( parttag )
351     pt = parttag.split(':')
352     pos = znajdz_pos(pt[0])
353     if pos
354         pt0=pt[0]
355         pt.delete_at( 0 )
356         atrybuty = pos.prawe
357         atryb0, atryb1= atrybuty.dx, atrybuty.dy
358         atryb0.each_index{ |i|
359             if self.colonate?( atryb0[i] )
360                 pt[i, 2] = pt[i,2].compact.join(':')
361             end
362             ##          puts [atryb0[i], pt0 ].inspect
363             if ( not pt[i] ) and x = self.preferred?( atryb0[i], pt0 )
364                 atryb0[i] = x
365             end
366         }
367     # atryb1 niesie informację o opcjonalności atrybutów.
368     ile_opc = atryb1.nitems
369     matches = [pt0]

372     skojarz = Proc.new {|i|
373         # ta proc bierze dotychczasową tablicę częściowych tagów i do każdego
        jej elementu dokłada (domnaża kartezjańsko) wszystkie możliwe
        wartości atrybutu i-tego, po czym wynik takiej operacji kompaktuje i
        spłaszcza.
374         znajdz_attr( atryb0[i] ).prawo.collect{ |a|
375             if a =~ Regexp.new(
376                 "^" + Regexp.escape( pt[i].to_s )
377                 # , "i" case-insetivity removed while fixing #356
                2010/6/16.
378                 ) # warunek spełniony także, gdy atrybut jest
                pusty (nil).

```

```

379         matches.collect{ |m| m + ':' + a }
380     end

382     }.compact.flatten}

384     0.upto([pt.size, atryb0.size - ile_opc].min) {|i|
385         # mnożymy kartezjańsko zbiory dopasowań kolejnych atrybutów
386         if atryb0[i] and not atryb1[i]
387             matches = skojarz.call( i )
388         end
389     }

391     ## puts atryb1.join(':')
393     ## puts atryb1.size
395     atryb1.size.times{ |i|
396         # dla atrybutów opcjonalnych (których więcej niż 1, mianowicie 2, ma
          tylko jedna klasa gramatyczna,
397         if atryb1[i] and ( pt[i] or (i-ile_opc >= 0 and pt[i-1# ile_opc
398                                     ])) # jeśli mamy
          opcjonalny lub następny
          jest opcjonalny – nie.
          Tylko jeśli mamy
          opcjonalny: 2008/9/23
          AP stwierdził, że w tej
          jednej klasie gram. drugi
          arg. opcjonalny może się
          pojawić tylko gdy
          występuje pierwszy –
          język tagsetu jest za
          słaby, żeby to wyrazić. Na
          razie postanowił(liśmy)
          nie rozbudowywać języka
          tagsetu o zagnieżdżenia
          opcjonalności, tylko
          zakodować to na twardo.

399         puts( i )
400         newmatches = skojarz.call(i)
401         if pt[i] : matches = newmatches
402         else matches += newmatches
403         end
404     end}

406     else # niepełna pos
407         re = Regexp.new("^#{Regexp.escape( parttag )}")
408             # , "i" case-insensitivity removed 2010/6/16 while fixing
              #356
409             )
410         matches = znajdz_pos( :all ).collect{ |t|
411             t.lewe if t.lewe.match re
412         }.compact
413     end

```

```

415     return matches# .sort nie sortujemy, żeby np. przypadki były w
        kolejności przypadków, a nie alfabetycznej – będą tak, jak są w pliku.
416 end # of self.matches

420 end# of class

```

### token.rb

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: token
5   #
6   # token_id :integer primary key
7   # kolejnosc :integer
8   # akapit_id :integer not null
9   # xpointer :text
10  # segmentation_xmlid :text
11  # morphosyntactic_xmlid :text
12  # fs_morph_comment :text
13  # path_id :integer not null
14  # orth :text
15  # czy_interp :boolean not null
16  # ns_poprzedza :boolean
17  # ns_nastepuje :boolean
18  # czy_konczy_zdanie :boolean
19  # czy_konczy_zdanie_updated_at :timestamp
20  # sg_choice_id :integer
21  # sg_variant_id :integer
22  # dodany :boolean not null
23  # chosen :boolean default(TRUE)
24  # chosen_updated_at :timestamp
25  # superancja :text
26  # created_at :timestamp
27  # updated_at :timestamp
28  #

52  require 'jcode'

54  class Token < ActiveRecord::Base
55    has_many :interpretacja, :order => "numer_lex_token,
        interpretacja_id",
56    :include => :leksem,
57    :conditions => [ " leksem.leksem_id not in ( "
        + Leksem.idi_nulli_et_ignoti.join(', ') +")" ]
58    # wołamy id dla Leksem.nullus, żeby zrobić się wyjątek, jeśli nullus nie
        istnieje: przekazanie nila na prawą stronę różności spowodowałoby
        zapewnienie, że zapytanie zwróci o rekordów.

60    has_many :interpretacja_export # widok do generowania wypłuwki
        (migr. 051)

62    has_many :interpretacja_wo_leksem, :class_name => "Interpretacja",

```

```

63      :order => "numer_lex_token, interpretacja_id"
64      # bo token.interpretacja.maximum( :nr_lex_token ) nie zadziałało: nie
      został dołączony leksem (migracja 038).

67      has_many :interpretacja_anot, :order => "akapit_transzy_id"
68      has_many :sens_anot
69      belongs_to :akapit
70      belongs_to :path

72      belongs_to :sg_variant
73      belongs_to :sg_choice

75      has_one :the_disambiguation, :class_name => "Interpretacja",
76      :conditions => {:disamb => true}, :include => :leksem

78      has_many :koniec_zdania_anot

80      serialize :superancja

82      def disambs( param )
83          # zwraca tablicę dyzambiguacyj (tablicę obiektów klasy Interpretacja).
84          # param może być idem użytkownika (gdy Integer) albo idem akapitu
          transzy (gdy hasz).
85          # Najpierw patrzymy, czy token jest zdezambiguowany – co możliwe
          tylko gdy wybór pewnej interpretacji został uzgodniony przez obie
          anotatorki. Jeśli token nie został zdezambiguowany, to
86          # szukamy interpretacji wyróżnionej przez daną anotatorkę (przez dany
          akapit transzy).
87          ##          @@poprz ||= Time.now
89          ##          @@nast ||= Time.now
91          ##          loguj = Proc.new { |w|
92              ##          @@poprz, @@nast = @@nast, Time.now
93              ##          logger.info "### tok.#{self.id}.disambs w.#{w}: #{@@nast -
          @@poprz}"
94          ##          }
95
100         # loguj.call( 65 )
101         if the_disa = self.the_disambiguation
102             return [ the_disa ]
103             ##                                     loguj.call( 66 )
104         else
105             ##                                     loguj.call( 68 )
106             return ( if param.kind_of?( Integer )
107                 phash = { :uzytkownik_id => param }
108             elsif param.kind_of?( Hash )
109                 # wariant z haszem dołożony 2008/09/01, żeby wyszukiwać po
110                 numerze akapitu transzy.
111                 phash = param.dup
112             else
113                 phash = nil
114             end
115
116             phash[ :token_id ] = self.token_id if phash
117
118             if phash

```

```

120      ##
      loguj.call( :przed_ia_find )
122      ##
      logger.info "### #{phash.inspect}"
124      ia=InterpretacjaAnot.find(
125          :first,
126          :conditions => phash )
127      ##
      loguj.call( :po_ia_find )
129      else
130          ia = nil
131      end
133      if ia : [ ia.interpretacja ]
134      else []
135      end )
136  end
138  end # of disamb.
140  def zambiguuj
141      logger.info "### ambiguujemy token #{self.id} »#{self.orth}«"
142      di = self.the_disambiguation
143      if di
144          di.disamb = false
145          di.save!
146      end
147  end
150  def do_sensu?( param )
151      # To musi być zrelatywizowane do anotatora,
152      # bo inny anotator mógł wybrać inną dyzambiguację,
153      # która nie ma leksemu osensowanego.
154      # Jeśli do sensu, to zwracamy id dyzambiguacji
155      ## logger.info( "do_sensu, param=:::#{param}:::" )
157      disas = self.disambs( param ) # na tym etapie anotacji jest
      zdezambiguowany.
158      if disas.size == 1
159          if disas[0].cz_m_leksem
160              # część mowy jest nie dla wszystkich określona, dlatego najpierw
              sprawdzamy, czy nie nil
161              return disas[0].id
162          else return false
163          end
164          else
165              return false
166          end
167  end # of do_sensu?( param )
170  def sensy( param )
171      # zwraca tablicę sensów: [(xmlid + short_def), sensy_id]
172      # dla sensów dotąd nie przypisanych zwraca [ ... , ... , -sensy_id ]
173      le187 = self.disambs( param )[0].cz_m_leksem # na tym poziomie
      anotacji jest zdezambiguowany

```

```

174     le187.sensy # one są posortowane po sensy_id — patrz cz_m_leksem.rb.
175     end # of sensy( param ).

178   def interp?( param )
179     it = ( self.disamb( param )[ 0 ] || self.interpretacja[0] )
180     if it
181       it.leksem.klasa_gram.klasa_gram_ozn == KlasaGram::INTERP
182     else nil
183     end
184   end

188   def the_sens( param = {} )
189     # jeśli sensu nie ma lub jest więcej niż 1 — zwraca nil.
190     # wpp. zwraca obiekt klasy SensyLeksemu.
191     # najpierw sprawdza, czy jest sens zaanotowany przez tego uza
192     # a potem z sensów leksemów.
193     sa187 = SensAnot.znajdz( self.id, param )

195     if sa187 : return sa187.sensy
196   else
197     s1s187 = self.disamb( param )[0]. # na tym poziomie anotacji jest
198     cz_m_leksem.sensy
199     if s1s187.size == 1
200       return s1s187[0]
201     else return nil
202     end
203   end
204   end # of the_sens( param ).

207   def xml_id
208     if self.original_id : return self.original_id
209     else return self.token_id.to_s
210     end
211   end

214   def <=>( drugi )
215     # z założenia będziemy porównywać tylko tokeny tego samego akapitu.
216     # porównania potrzebujemy do posortowania listy potencjalnych głów
217     frazy
218     if drugi.respond_to?(:nr_wyrazu_zd)
219       drid = drugi.nr_wyrazu_zd.to_i
220     else drid = drugi.to_s.to_i
221     end
222     return( self.nr_wyrazu_zd.to_i <=> drid )
223   end

225   def lemat( param )
226     di = ( self.disamb( param )[0] || self.interpretacja[0] )
227     di.leksem.lemat if di
228   end

232   def najbliższy( minmax, inequal, ponad_akapity=nil )

```



```

233     # 2010/2/24 w pełni dualna do poprzedni.
235     najbl = nil
237     unless self.chosen? # we're not chosen. That means we are in a
        segmentation variant (our sg_variant_id is not null). So first we try
        return previous token from our variant and if there's no such then from
        other chosen tokens.
238         najbl = Token.find( :first, :conditions =>
239             [ " kolejnosc = ( select #{minmax}( kolejnosc )
                from token " +
240                 " where kolejnosc#{inequal}#{self.kolejnosc}
                and " +
241                 " akapit_id=#{self.akapit_id} and " +
242                 " sg_variant_id=? )", self.sg_variant_id ]
243             )
244     end
246     unless najbl
247         najbl = Token.find( :first, :conditions =>
248             [ " kolejnosc=(select #{minmax}(kolejnosc) from
                token " +
249                 " where kolejnosc#{inequal}#{self.kolejnosc}
                and " +
250                 " ( sg_choice_id<>? or sg_choice_id is null )
                and " +
251                 " akapit_id=#{self.akapit_id} and chosen='t'
                )", self.sg_choice_id ]
252             )
253     end
255     if ponad_akapity and not najbl
256         # potrzebujemy tego do poprawienia xpointerów. Wówczas patrzymy
        na xpointery poprzedniego i następnego tokenu oryginalnego, czyli
        dodany=false.
257         najbl = Token.find( :first, :conditions =>
258             [ " kolejnosc=(select #{minmax}(kolejnosc) from
                token " +
259                 " where kolejnosc#{inequal}#{self.kolejnosc}
                and " +
260                 " path_id=#{self.path_id} and dodany='f' )"
                # poprzedniość/następność nie ma sensu między
                różnymi ścieżkami Korpusu.
261             )
262     end
264     najbl
265 end # of najbliższy
268 def poprzedni( ponad_akapity=nil )
269     self.najbliższy( "max", "<", ponad_akapity )
270 end
273 def następny( ponad_akapity=nil )

```

```

274     self.najblizszy( "min", ">", ponad_akapity )
275 end

278 alias :nastepny :następný

281 def odsmiec_interpretacje
282   Interpretacja.odsmiec( :token_id => self.id )
283 end# of odsmiec_interpretacje.

286 def sg_variant_anot( akathasz )
287   if sgvid = self.sg_variant_id
288     SgVariantAnot.find( :first, :conditions =>
289       akathasz.dup.update( :sg_variant_id => sgvid ))
290   else nil
291   end
292 end # of sg_variant_anot.

295 def segmentny?( akathasz )
296   self.chosen and (
297     ( not ( sgvan = self.sg_variant_anot( akathasz ) ) )
298     or
299     sgvan.chosen? )
300 end# of segmentny?

302 def self.update_ns_nastepuje( ## akapit_od, akapit_do,
303                               force=nil )
304   if self.find_by_sql(
305     "select 1 as val from token where dodany='f' and "
306     +
307     " ns_nastepuje is null limit 1" ) [0]
308     coś_jest_na_rzeczy = true
309   else
310     coś_jest_na_rzeczy = false
311   end

313   if force ## or not self.find(:first, :conditions => {
314     ## :ns_nastepuje => true, :akapit_id => akapit_od..akapit_do } )
315     zawężenie = ""
316   else
317     zawężenie = " and ns_nastepuje is null "
318   end

322   # 2009/10/8 nie jest dla mnie jasne, czy poniższe SQL-e to nie jest
323   # stąpanie po kruchym lodzie: a co jeśli nps przed wariantem
324   # segmentacyjnym?
325   self.connection.execute " update token set ns_nastepuje='t' " +
326     " where token_id in " +
327     " ( select token_id-1 from token t1 where t1.ns_poprzedza='t' ) "
328   +
329   " and dodany='f' " + zawężenie
330   ## " and akapit_id between #{akapit_od} and #{akapit_do} "
331   # jeszcze kwestia źle przypisanych nps-ów:
332   self.connection.execute " update token set ns_nastepuje='f' " +

```

```

331     " where token_id in " +
332     " ( select token_id-1 from token t1 where t1.ns_poprzedza='f' ) "
333     +
334     " and dodany='f' " + zawężenie
335     ##      " and akapit_id between #{akapit_od} and #{akapit_do} "
336     if zły_t = self.beginning_ns_poprzedza[0]
337         ## raise "detected token with 'nps' attribute beginning a
338         choice: #{zły_t.id} »#{zły_t.orth}«"
339     end
340     return ( coś_jest_na_rzeczy or force )
341 end # of self.update_ns_następuje

342 def self.beginning_ns_poprzedza
343     self.find( :all, :conditions => "ns_poprzedza='t' and token_id in
344     (select min(token_id) from token t1 where sg_variant_id is not
345     null group by sg_variant_id)")
346 end # of self.beginning_ns_poprzedza

347 ## update_ns_następuje

348 def lex_msd
349     # Warning: this method uses the interpretacja_export view so skips
350     the interpretations that are manually added and are not
351     disambiguations.
352     # zwraca tablicę tablic interpretacji pogrupowanych podług leksemów.
353     intex = self.interpretacja_export
354
355     lexids_u = intex.collect { |int199| int199.leksem_id }.uniq # są
356     w odp. kolejności
357
358     igr199 = intex.group_by { |int199| int199.leksem_id }
359     # tu interpretacje exportowe grupujemy podług leksem_id.
360
361     lexes = lexids_u.collect { |lid199|
362         igr199[ lid199 ]
363     } # a tu zamieniamy hasz na tablicę.
364
365     return lexes
366 end # of lex_msd

367 def self.uniq_superancje!
368     self.find( :all, :conditions => "superancja is not null" ).each {
369         |tok|
370         tok.superancja.uniq!
371         tok.save!
372     }
373
374     true
375 end # of self.uniq_superancje!

376 def superancja_include?( poziom )
377     (ssu = self.superancja) and ssu.include?( poziom )
378 end

379 def self.update_kolejnosc

```

```

387     if self.find_by_sql( "select * from token where kolejnosc is null
388       limit 1" )[0]
389         self.connection.execute "update token set kolejnosc =
390           token_id*216 where kolejnosc is null"
391     end
392   end
393
394   update_kolejnosc
395
396   def odzółć( poziom )
397     if self.superancja
398       self.superancja -= [ poziom ].flatten
399       self.superancja = nil unless self.superancja[0]
400       self.save!
401     end
402   end # of odzółć
403
404   def choice_card
405     self.class.find_by_sql(
406       [ "select count(*) as card from sg_variant
407         where sg_choice_id=?",
408         self.sg_choice_id ] )[0].card.to_i
409   end
410
411   def sg_possies
412     # możliwości dla „wybierz/odrzuć ten wariant segmentacyjny”: jeśli
413     jesteśmy w co najmniej trójce wariantów, to nie możemy odrzucać, bo by
414     to nie było jednokrokowe.
415     if self.choice_card <= 2
416       [ true, false ]
417     else [ true ]
418     end
419   end
420
421   def self.gsub_SHY
422     sth_done211 = false
423     akids211 = []
424     self.find( :all, :conditions => { :orth => '-' } ).each { |t211|
425       t211.orth = t211.orth.gsub( '-', '-' )
426       t211.save!
427       akids211 << t211.akapit_id
428     }
429
430     puts akids211.inspect
431
432     Akapit.find( akids211.uniq ).each{ |aka211|
433       aka211.update_treści
434       sth_done211 = true unless sth_done211
435     }
436
437     return sth_done211
438   end # of self.gsub_SHY
439
440   def self.sprawdź_xpointer( id )

```

```

442     t = self.find( id )
443     if t.sg_variant_id
444         tt = t.sg_variant.token
445         tt = [ tt[0].poprzedni(:ponad_akapity) , tt, tt[-1].nastepny(
:ponad_akapity ) ].flatten
446     else
447         tt = [t.poprzedni( :ponad_akapity ), t, t.nastepny(
:ponad_akapity ) ]
448     end
449
450     tt.each { |t1|
451         if t1
452             puts "#{t1.id} #{if t1.ns_poprzedza? : '<ns>' else '<spacja>'
end} " +
453                 " #{t1.orth} #{if t1.ns_nastepuje : '<ns>' else '<spacja>'
end} #{t1.xpointer}" +
454                 " (sgv #{t1.sg_variant_id.inspect})"
455         else
456             puts "<nil>"
457         end
458     }
459
460     nil
461 end
462
463 def self.ustaw_xpointer_na( tokid, xpo )
464     t1 = self.find( :first, :conditions => "token_id=#{tokid}" )
465     if t1 and t1.xpointer != xpo
466         t1.xpointer = xpo
467         t1.save!
468     end
469 end # of self.popraw_xpointer
470
471 def self.popraw_xpointery_nullowe
472     sth_done = false
473     self.find( :all, :conditions => "xpointer is null" ).each { |t|
474
475         sth_done = true unless sth_done
476
477         tpo = t.poprzedni( :ponad_akapitami )
478
479         if tpo
480             md = /^(.*) ,(\d+),(\d+)\)$/ .match( tpo.xpointer )
481
482             md_poprz = [ md[ 1 ] + ', ', # dodajemy przecinek, bo xpointery
483                         mają postać „...ab,nr,nr”.
484                         md[ 2 ].to_i, md[3].to_i ]
485
486             t.xpointer = md_poprz[0] + "#{md_poprz[1]+md_poprz[2]+if
487 t.ns_poprzedza? : 0 else 1 end},#{t.orth.length}" + ')'
488
489         else # the previous token does not exist. Then the next one must exist
490             since paragraphs are about 50-word.
491             md = /^(.*) ,(\d+),(\d+)\)$/ .match( t.nastepny(
:ponad_akapitami ).xpointer )

```

```

492         md_nast = [ md[ 1 ] + ', ', # dodajemy przecinek, bo xpointery
                    mają postać „...ab,nr,nr”.
493         md[ 2 ].to_i, md[3].to_i ]

495         t.xpointer = md_nast[0] + "#{md_nast[1]-(t.orth.jlength+if
                    t.ns_nastepuje? : 0 else 1 end)},#{t.orth.jlength}" + ')'

497     end

499     t.save!

501     puts "poprz: #{if t.poprzedni : t.poprzedni.xpointer end}
          id: #{t.id} #{t.ns_poprzedza.inspect} #{t.orth} #{t.xpointer}
          nast: #{if t.nastepny : t.nastepny.xpointer end}"

503   }

505   sth_done

507   end # of self.popraw_xpointery_nullowe

511   end # of class.

514   ## # Local Variables:
515   ## # mode: ruby
516   ## # End:

transza.rb

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: transza
5   #
6   # transza_id :integer primary key
7   # uzytkownik_id :integer
8   # created_at :timestamp
9   # updated_at :timestamp
10  #

34  require 'ramkowanie'

36  Uzytkownik.find( :first ) # just to wake it up

38  class Transza < ActiveRecord::Base

40    LICZBA_AKAPITÓW = 2 * AnoVersion.rozmiar_transzy
41    # klasa AnoVersion jest zdefiniowana w pliku nk8003.rb.

43    belongs_to :uzytkownik

45    has_many :akapit_transzy, :order => "status desc, akapit_transzy_id"

47    has_many :akapity_transzy_niezatwierdzone,
48      :class_name => "AkapitTranszy",
49      :order => "status desc, akapit_transzy_id",
50      :conditions => [" status < ? ", AkapitTranszy::STATI[:zatwierdzony]]

52    has_many :akapity_transzy_zatwierdzone,

```

```

53      :class_name => "AkapitTranszy",
54      :order => "status desc, akapit_transzy_id",
55      :conditions => [" status = ?", AkapitTranszy::STATI[:zatwierdzony]]

57      has_many :akapity_transzy_rozbiezne,
58      :class_name => "AkapitTranszy",
59      :order => "status desc, akapit_transzy_id",
60      :conditions => [" status between ? and ? ",
61                      AkapitTranszy::STATI[:zatwierdzony]+1,
62                      AkapitTranszy::STATI[:zweryfikowany]-1]

64      has_many :akapity_transzy_zamkniete,
65      :class_name => "AkapitTranszy",
66      :order => "status desc, odrzucony, akapit_transzy_id",
67      :conditions => ["status >= ?", AkapitTranszy::STATI[:zweryfikowany]]

69      has_one :prosby_annotatorek, :as => :dotyczy

71      if DlaEli.x
72        MAX_TRANSZ = 5
73      else MAX_TRANSZ = 25 # podwyższone z 5 2009/7/4 (#162)
74      end

77      def validate
78        # musimy sprawdzać, że użytkownik nie ma dwa razy takiej samej
79        # np. ustanawiając numerację transz od 0,
80        # że (nr_nowej div 2) not in select (nry_przydzielonych div 2)
81      end

83      def opis
84        ile_zaa = Proc.new { |* status_sym|
85          AkapitTranszy.ile_zaanotowanych({:transza_id => self.id}, *
86            status_sym )
87        }
88        zakończonych = ile_zaa.call( :zakończony )
89        oczekujących = ile_zaa.call( :oczekujące )
90        anotaśnych = ile_zaa.call( :anotaśne, :to_ary )
91        if anotaśnych.max == 0
92          do_annotacji = "nic do anotacji"
93        else
94          do_annotacji = " #{anotaśnych.inspect} do anotacji"
95        end

96        podsądnych = ile_zaa.call( :podsądny, :to_ary )
97        podsądnych =
98          if podsądnych.max == 0
99            "
100          else
101            " i #{podsądnych.inspect} pod sądem"
102          end

104        "#{self.transza_id}" +

```

```

105      " (#{self.ile_akapitów} ak., w tym #{zakończonych} zak. i
      #{oczekujących} ocz. – #{do_anotacji}#{podsądnych})"
106  end # of opis

108  alias :opis_nohtml :opis

110  def ile_akapitów # myślę, że zadziała dużo szybciej i prościej niż
  self.akapit_transzy.size.
111    self.class.find_by_sql(
112      [ "select count(*) as ile from akapit_transzy
        where transza_id=?",
        self.id ] ) [0].ile.to_i
113
114  end

116  def sa_jakies_rozbieznosci?( over_symb )
117    saro=AkapitTranszy.find_by_sql([
118      "select count(*) as ile from
        akapit_transzy " +
119      " where transza_id = ? and status
        between ? and ? ",
120      self.transza_id,
121      AkapitTranszy::STATI[:zatwierdzony]+1,
122      AkapitTranszy::STATI[over_symb]-1,
123      ])[0].ile.to_i
124    if saro==0 : return nil ; else return saro end
125  end

128  def do_poprawki?
129    sa_jakies_rozbieznosci?( :po_konfrontacji2 )
130  end

133  def sa_rozbiezne?
134    sa_jakies_rozbieznosci?( :zweryfikowany )
135  end

138  def self.lista_transz( meth_symb = :wsze_transze )
139    # zwraca tablicę par [ <anotarka>, [<jej transze (np. otwarte)]]
140    anotatores = Uzytkownik.find(:all,
141      :conditions => {:rola_id => Rola.rid( :anotator
142      )},
143      :order => :login)
144    if meth_symb == :wsze_transze
145      tros = anotatores.collect{ |anor| [anor,
146        anor.transze_otwarte + ( anor.transza
147        - anor.transze_otwarte )] }
148    else
149      tros = anotatores.collect{ |anor| [anor, anor.send(meth_symb)] }
150    end
151    return tros
152  end # of self.lista_transz.

153  def self.wsze_otwarte
154    self.lista_transz( :transze_otwarte )

```



```

155     end
157     def self.wsize
158         self.lista_transz
159     end
161     HOW_MANY_RECENT = 40
163     def self.recent_closed
164         # we'll return the list of how_many most recently closed transzas.
165         self.find_by_sql ( "select x.*, login from (select transza.*,max(
max(a.updated_at), max( s.updated_at )) as true_updated_at from
transza inner join akapit_transzy a using( transza_id )inner join
statusy s using( akapit_transzy_id )where (select count(*) from
akapit_transzy psi where
psi.transza_id=transza.transza_id)=(select count(*) from
akapit_transzy ksi where ksi.transza_id=transza.transza_idand
ksi.status>=16 )group by transza.transza_idorder by
true_updated_at desc limit #{HOW_MANY_RECENT}) as xinner join
uzytkownik using( uzytkownik_id )order by login, true_updated_at
desc;"
166     )
167     end
171     def self.zrob_transze( pełne=nil )
172         id_null = "select akapit_id from akapit a left outer join
akapit_transzy at using( akapit_id ) " +
173         " where akapit_transzy_id is null "
174         if self.find_by_sql( " select 1 as jeden where exists (
#{id_null } ) " ) [0]
175             if pełne
176                 inu = self.find_by_sql( id_null ).collect{ |a| a.akapit_id }
177                 id_null = inu[ 0, [inu.size - inu.size % LICZBA_AKAPITÓW,
0].max ]
178                 cond = { :akapit_id => id_null }
179             else
180                 cond = " akapit_id in ( #{id_null } ) "
181             end
182             a = Akapit.find( :all, :conditions => cond ) # ostatni warunek
jest prawie na pewno redundantny, tzn. ograniczenie »od-do«.
183             as200=a.size/LICZBA_AKAPITÓW+1
184             as200.times{ |i|
185                 akaa = a[i* LICZBA_AKAPITÓW, LICZBA_AKAPITÓW]
186                 pocwiartuj( akaa )
187             }
189         end# of main unless
191         self.unnull_akat_bliźniaki
193     end# of self.zrob_transze.
195     private
197     def self.zrobakapitytranszy( akaa_n )

```

```

198     if akaa_n[0]
199       # 2009/05/22, #116 na Tracu: wersja 8005 miała transze puste (bo w I
        turze wiano 17 akapitów.
200       tra = self.create
201       akaa_n.each {|aka| tra.akapit_transzy.create( :akapit_id =>
        aka.akapit_id ) }
202     end
203   end

206   def self.pocwiartuj( akaa )
207     # np. dla tablicy 200 akapitów dzielimy je po 50
208     # i łączymy 0: I i II, 1: III i IV, 2: I i III, 3: II i IV
209     cwierc=akaa.length/4
210     akaa= [
211       akaa[0,cwierc* 2],
212       akaa[cwierc* 2, cwierc* 2+3],
213       akaa[0, cwierc] + akaa[cwierc* 2, cwierc],
214       akaa[cwierc, cwierc] + akaa[cwierc* 3, cwierc+3]
215     ]

217     self.zrobakapitytranszy( akaa[0] )
218     self.zrobakapitytranszy( akaa[1] )
219     self.zrobakapitytranszy( akaa[2] )
220     self.zrobakapitytranszy( akaa[3] )
221   end# of self.pocwiartuj.

223   ##   zrob_transze
225   ##   Statusy.find( :first )

228   end # of class.

230   ## # Local Variables:
231   ## # mode: ruby
232   ## # End:

```

### uzytkownik.rb

```

1   # == Schema Information
2   # Schema version: 51
3   #
4   # Table name: uzytkownik
5   #
6   # uzytkownik_id :integer primary key
7   # rola_id :integer not null
8   # login :text not null
9   # hasz_haslo :text not null
10  # sol :text not null
11  # imie :text
12  # nazwisko :text
13  # email :text
14  # jak_anotator :boolean
15  # created_at :timestamp
16  # updated_at :timestamp

```

```
17 # haslo_updated_at :timestamp
18 # tylko_nowe_komentarze_ogladactwo :boolean default(TRUE)
19 # tylko_nowe_komentarze_annotacja :boolean
20 #

44 require 'jcode' # bo rake annotate_models się wywalało na jlength
45 require 'digest/sha1'

47 class Uzytkownik < ActiveRecord::Base

49   # wydaje się nie być technicznych przeciwwskazań do łączenia tej samej
   bazy użytkowników do dwóch wersji Anotatorni naraz, ale wydaje się być
   przeciwwskazanie merytoryczne (np. stara Anotatornia ma „śmieciowych”
   nat1-3, których nowa jako żywo nie potrzebuje.

51   ActiveRecord::Base.connection.execute "attach database
   \"uzytkownicy_{AnoVersion.port_uzy}.db\" as \"uzy\""

53   belongs_to :rola

55   has_many :transza
56   has_many :transze_otwarte, :class_name => "Transza",
57   :conditions => ['exists (select * from akapit_transzy akatwhere
   akat.transza_id = transza.transza_id and akat.status < ? )'],
58   AkapitTranszy::STATI[ :zweryfikowany ]]

60   has_many :transze_zamkniete, :class_name => "Transza",
61   :conditions => ['not exists (select * from akapit_transzy akatwhere
   akat.transza_id = transza.transza_id and akat.status < ? )'],
62   AkapitTranszy::STATI[ :zweryfikowany ]]

65   has_many :interpretacja_anot
66   has_many :sens_anot

68   has_many :odebrania_transz, :class_name => "ProsbyAnotatorek",
69   :conditions => {:dotyczy_type => "Transza"}, :order => " updated_at
   desc, created_at desc "

71   has_many :odebrania_odebrane, :class_name => "ProsbyAnotatorek",
72   :conditions => {:dotyczy_type => "Transza", :spelniona => true}

74   has_one :poziomy_annotacji

76   validates_presence_of :login
77   validates_uniqueness_of :login

79   attr_accessor :haslo_confirmation

81   validates_confirmation_of :haslo
82   validates_inclusion_of :rola_id, :in => Rola.find(:all).collect
   {|rola| rola[:rola_id]}

84   validates_presence_of :email
85   validates_format_of :email, :with => %r{[a-z].* @[a-z].* \.
   [a-z]}i,
86   :message => "musi zawierać literę, @, literę, kropkę i literę."

88   def validate
```

```

89 # errors.add_to_base("Brakuje hasla" ) if @haslo.blank?
90   if @haslo # bo nie zawsze uaktualniamy uzytkownika z haslem.
91     errors.add_to_base("Haslo musi miec co najmniej 8 znakow") if
      @haslo.length < 8
92     # require 'jcode' jest w application.rb
93     errors.add_to_base("Haslo musi zawierac wielka litere") unless
      @haslo =~ /[A-Z]/
94     errors.add_to_base("Haslo musi zawierac mala litere") unless
      @haslo =~ /[a-z]/
95     errors.add_to_base("Haslo musi zawierac cyfre") unless @haslo
      =~ /[0-9]/
96   end
97 end
98 # I could have written the above with the helper methods
      validates_length_of and validates_format_of

100 def self.authenticate(login, haslo)
101   uzytkownik = self.find_by_login(login)
102   if uzytkownik
103     haslo_spodziewane = haslo_szyfrowane( haslo, uzytkownik.sol )
104     if uzytkownik.hasz_haslo != haslo_spodziewane
105       uzytkownik = nil
106     elsif uzytkownik.rola_id == Rola.rid( :zablokowany )
107       uzytkownik = :zablokowany
108     end
109   end
110   uzytkownik
111 end

113 # 'password' is a virtual attribute
114 def haslo
115   @haslo
116 end

118 def stare_haslo
119   # tylko po to, żeby tworzył się formularz zmiany hasła
120 end

122 def haslo=(pwd)
123   @haslo = pwd
124   zrob_nowa_sol
125   self.hasz_haslo = Uzytkownik.haslo_szyfrowane(self.haslo,
      self.sol)
126 end

128 def after_create
129   PoziomyAnotacji.nadaj_zu_uzytkownikiem
130 end

132 def after_destroy
133   if Uzytkownik.find(:all, :conditions => {:rola_id =>
      Rola.rid(:admin) }).size.zero?
134     raise "Nie usunę ostatniego admina."

```

```
135     else
136       PoziomyAnotacji.nadąż_za_użytkownikiem
137     end
138   end

140   def after_update
141     if Uzytkownik.find(:all, :conditions => {:rola_id =>
142       Rola.rid(:admin) }).size.zero?
143       raise "Nie zablokuję ostatniego admina."
144     end
145   end

147   def anotator?
148     Rola.ktoras_z_rol?(self, :anotator)
149   end

151   def zarzadca?
152     Rola.ktoras_z_rol?(self, :zarzadca)
153   end

155   def ktoras_z_rol?( * role )
156     Rola.ktoras_z_rol?(self, * role )
157   end

159   def transza_ids
160     self.transza.collect{ |tra| tra.id }
161   end

163   def ile_zakończonych
164     AkapitTranszy.ile_zaanotowanych( {:transza_id =>
165       self.transza_ids}, :zakończony )
166   end

167   def ile_oczekujących
168     AkapitTranszy.ile_zaanotowanych( {:transza_id =>
169       self.transza_ids}, :oczekujące )
170   end

171   def ile_anotaśnych
172     AkapitTranszy.ile_zaanotowanych( {:transza_id =>
173       self.transza_ids}, :anotaśne )
174   end

176   def no_record?( tab_symb )
177     uid = self.uzytkownik_id
178     if tab_symb == :statusy
179       where = PoziomyAnotacji.poziomy_x.collect {|poz|
180         " #{poz}_uzid = #{uid} "}.
181       join(" or " )
182     end

183     else
184       where = " uzytkownik_id = #{uid} "
185     end

187     tab_symb.to_s.camelize.constantize.find_by_sql(
```

```

188                                     "select count(*) as cnt
189                                     from #{tab_symb.to_s} " +
190                                     " where #{where}"
191                                     )[0].cnt.to_i == 0
192
193     end # of no_record?
194
195     def self.natror
196         self.find_by_login( 'natror' )
197     end
198
199     def ile_transz_zakończonych( poziom = :max )
200
201         where_poziom202 =
202             if poziom == :max
203                 PoziomyAnotacji.maksymalne & PoziomyAnotacji.działające
204             else
205                 [ poziom ]
206             end .
207             collect { |po202|
208                 "#{po202}<=#{Statusy::STATI[ :po_poprawce ]}"
209             }.join( ' or ' ) # wszystko musi być na odwrót (po de Morganowsku),
210                             # bo idzie do klauzuli not exists.
211
212         Transza.count_by_sql(
213             "select count(*) from transza where
214             uzytkownik_id=#{self.id}" +
215             " and not exists (select 1 from statusy s where "
216             +
217             " s.transza_id=transza.transza_id and " +
218             " #{where_poziom202})"
219         )
220
221     end # of ile_transz_zakończonych
222
223     private
224
225     def self.haslo_szyfrowane(haslo, sol)
226         string_to_hash = haslo + "wyswi17" + sol # 'wibble' makes it harder
227         to guess
228         Digest::SHA1.hexdigest( string_to_hash )
229     end
230
231     def zrob_nowa_sol
232         self.sol = self.object_id.to_s + rand.to_s
233     end
234
235     def self.natrorize
236         if UNatrora.x
237             self.find(:all,
238                 :conditions => " login not like 'natror%' and login
239                 <>'QueenOfSpades' "
240                 ).each{ |uzytkownik|
241                     haslo= uzytkownik.login + '666aA'
242                     uzytkownik.haslo = haslo

```

```

241      uzytkownik.haslo_confirmation=haslo
242      if uzytkownik.save!
243          ##          puts "Zmieniłam hasło użytkownika
                #{uzytkownik.login} na '#{haslo}'."
245      end
246  }
247      puts "Zmieniłam hasła użytkowników na »<login>666aA«"
248  end # of if UNatrora.x
249  end # of self.natrорize

251  # if we are in a virgin database, we create a dummy admin to allow
252  # new licensee to add their own users.
253  if self.count == 0
254      self.create!(
255          :login => "QueenOfSpades",
256          :hasz_haslo =>
                '1467fdaf0fe9be193e02f053cd39788224d2741e',
257          # "15 mars 1615 Paris"
258          :sol => '-6098396180.298015019238499',
259          :imie => 'Marguerite',
260          :nazwisko => 'de France et Navarre',
261          :email => 'margot@louvre.fr',
262          :rola_id => Rola.rid( :admin ),
263          :tylko_nowe_komentarze_ogladactwo => true,
264          :tylko_nowe_komentarze_anotacja => true,
265          :jak_anotator => false
266      )
267  end

269  natrорize if UNatrora.x
270  # make_natrors removed for security reasons to anotatornia-trash/natrory.rb
    (not available in this version). Equivalent (and also not available here) SQL
    inserts are in anotatornia-trash/natrory.sql.
271  PoziomyAnotacji.nadaż_za_użytkownikiem
272  true

275  end # of class

279  ## # Local Variables:
280  ## # mode: ruby
281  ## # End:

```

### word\_senses\_rozbieznosc.rb

```

1  # == Schema Information
2  # Schema version: 51
3  #
4  # Table name: word_senses_rozbieznosc
5  #
6  # word_senses_rozbieznosc_id :integer primary key
7  # token_id :integer not null
8  # sensy_id :integer not null
9  # created_at :timestamp

```

```

10  # updated_at :timestamp
11  #

35  class WordSensesRozbieznosc < ActiveRecord::Base

37    has_one :punkt_protokolu, :as => :elt_protokol
38    belongs_to :token
39    belongs_to :sensy

41  end

```

## app/views/admin

### dodaj\_uzytkownika.rhtml

```

24  <div id="main">
25    <%= render :partial => '/layouts/notice' -%>

27  <h1>Logowanie &#8212; dodaj użytkownika</h1>
28  <%= error_messages_for 'uzytkownik' %>
29  <fieldset>
30    <legend>Wprowadź dane użytkownika</legend>
31    <% form_for :uzytkownik do |form| %>
32    <p>
33      <label for="uzytkownik_rola_id">Rola:</label>
34      <br/>
35      <%=
36        form.select :rola_id,
37          (Rola.find(:all, :order => :rola_id)).collect {|rola|
38            [rola[:opis], rola[:rola_id]]
39          },
40      :prompt => "Wskaż rolę użytkownika"
41      %>
42    </p>
43    <p>
44      <label for="uzytkownik_login">Login:</label>
45      <br/>
46      <%= form.text_field :login, :size => 40 %>
47    </p>
48    <p>
49      <label for="uzytkownik_haslo">Hasło:</label>
50      <br/>
51      <%= form.password_field :haslo, :size => 40 %>
52    </p>
53    <p>
54      <label for="uzytkownik_haslo_confirmation">Powtórz
55        hasło:</label>
56      <br/>
57      <%= form.password_field :haslo_confirmation, :size => 40 %>
58    </p>
59  </p>

```



```

59     <label for="uzytkownik_imie" >Imię użytkownika:</label>
60     <br/>
61     <%= form.text_field :imie, :size => 40 %>
62     </p>
63     <p>
64         <label for="uzytkownik_nazwisko" >Nazwisko użytkownika:</label>
65         <br/>
66         <%= form.text_field :nazwisko, :size => 40 %>
67     </p>
68     <p>
69         <label for="uzytkownik_email" >email użytkownika:</label>
70         <br/>
71         <%= form.text_field :email, :size => 40 %>
72     </p>
73     <%= submit_tag "Dodaj użytkownika" , :class => "submit" %>
74     <% end %>
75 </fieldset>
76 </div>

```

### lista\_uzytkownikow.rhtml

```

24 <div id="main">
25 <%= render :partial => '/layouts/notice' -%>
27 <h1>Lista użytkowników</h1>
28 <ul>
29     <% for uzytkownik in @wszyscy_uzytkownicy %>
30     <li>
31         <%= link_to "[zmień hasło]" , { # link_to options
32             :controller => 'admin',
33             :action => 'zmien_haslo',
34             :id => uzytkownik,
35             { # html options
36                 :method => :get,
37                 :confirm => "Czy rzeczywiście zmienić hasło dla
38                     #{uzytkownik.login}?"
39             }
40         %>
41     </li>
42 <% if Rola.usuwalny?(uzytkownik)
43     <%= link_to "[usuń]" , { # link_to options
44         :controller => 'admin',
45         :action => 'usun_uzytkownika',
46         :id => uzytkownik,
47         { # html options
48             :method => :post,
49             :confirm => "Czy rzeczywiście usunąć #{uzytkownik.login}?"
50         }
51     </li>
52 <% end
53 %>
55 <%=

```

```

56         link_to "[zablokuj]" , { # link_to options
57         :controller => 'admin',
58         :action => 'zablokuj_uzytkownika',
59         :id => uzytkownik},
60         { # html options
61         :method => :post,
62         :confirm => "Czy rzeczywiście zablokować #{uzytkownik.login}?
        (Jeśli to anotator, wszelkie nie zakończone transze
        zostaną mu/jej odebrane.)"
63     }
64 %>

66     <%= h(uzytkownik.login) + " (" + uzytkownik.rola.opis_kr+" )" +
67     if uhau = uzytkownik.haslo_updated_at
68     ', hasło zm. ' + uhau.to_s( :db )
69     else "
70     end -%>
71     </li>
72     <% end %>
73 </ul>
74 </div>

```

### usun\_uzytkownika.rhtml

```

24 <h1>Logowanie#usun_uzytkownika</h1>
25 <p>Find me in app/views/logowanie/usun_uzytkownika.rhtml</p>

```

### zablokuj\_uzytkownika.rhtml

```

24 <h1>Blokowanie użytkownika</h1>
25 <ul>
26   <% uzytkownik = Uzytkownik.find(params[:id]) %>
27   <li><%= link_to "[zablokuj]" , { # link_to options
28   :controller => 'logowanie',
29   :action => 'zablokuj_uzytkownika',
30   :id => uzytkownik},
31   { # html options
32   :method => :post,
33   :confirm => "Czy rzeczywiście zablokować #{uzytkownik.login}?"
34   } %>
35   <%= h(uzytkownik.login) + " (" + uzytkownik.rola.opis_kr+" )" %>
36   </li>
37 </ul>

```

### zmien\_dane.rhtml

```

24 <div id="main">
25   <%= render :partial => '/layouts/notice' -%>
26   <h1>Zmiana danych użytkownika <%= @uzytkownik.login%></h1>

29   <%= error_messages_for 'uzytkownik' %>
30   <fieldset>

```

```

31     <legend>Zmiana danych użytkownika <%= @uzytkownik.login%> </legend>
32     <% form_for :uzytkownik do |form| %>
33     <p>
34         <label for="uzytkownik_imie" >Imię użytkownika:</label>
35         <br/>
36         <%= form.text_field :imie, :size => 40 %>
37     </p>
38     <p>
39         <label for="uzytkownik_nazwisko" >Nazwisko użytkownika:</label>
40         <br/>
41         <%= form.text_field :nazwisko, :size => 40 %>
42     </p>
43     <p>
44         <label for="uzytkownik_email" >email użytkownika:</label>
45         <br/>
46         <%= form.text_field :email, :size => 40 %>
47     </p>
48     <%= submit_tag "Zmień dane użytkownika" , :class => "submit" %>
49     <% end %>
50 </fieldset>
51 </div>

```

## zmien\_haslo.rhtml

```

24 <div id="main">
25 <%= render :partial => '/layouts/notice' -%>
27 <h1>Zmiana hasła <%=
28     if session[:rola_id] & Rola.rid(:admin) != 0
29         "użytkownika <b>#{@uzytkownik.login}</b>"
30 end %></h1>
32 <%= error_messages_for 'uzytkownik' %>
33 <fieldset>
34     <legend>Zmiana hasła użytkownika <b><%=
35         @uzytkownik.login%></b></legend>
36     <% form_for :uzytkownik do |form| %>
37
38     <% if session[:rola_id] & Rola.rid(:admin) == 0 %>
39     <p>
40         <label for="uzytkownik_stare_haslo" >Stare hasło:</label>
41         <br/>
42         <%= form.password_field :stare_haslo, :size => 40 %>
43     </p>
44     <% end%>
45     <p>
46         <label for="uzytkownik_haslo" >Nowe hasło:</label>
47         <br/>
48         <%= form.password_field :haslo, :size => 40 %>
49     </p>

```

```

50      <label for="uzytkownik_haslo_confirmation">Powtórz nowe
      hasło:</label>
51      <br/>
52      <%= form.password_field :haslo_confirmation, :size => 40 %>
53      </p>
54      <%= submit_tag "Zmień hasło" , :class => "submit" %>
55      <% end %>
56    </fieldset>
58  </div>

```

## app/views/anotacja

### \_akapity\_skomentowane.rhtml

```

24  <% # partial renderowany z widoku ogladactwo
25  if r_wszewid? -%>

27  <h1>Akapity z <%= if session[:tylko_nowe_komentarze] : "nowym "
    end%>komentarzem</h1>

29  <% if session[:skomentowane_pokaż] %>

31  <%= link_to_remote '[schowaj]', :url => { :action =>
    :skomentowane_przełącz }
32  -%>

35  <% wsze_lub_nowe_hasz = { :url => { :action =>
36  :skomentowane_tylko_nowe_przełącz } }
37  unless (not session[:tylko_nowe_komentarze]) or true
38  wsze_lub_nowe_hasz[ :confirm ] = "Czy rzeczywiście chcesz
    oglądać#{Komentarz.ile} komentarz/e/y do #{Komentarz.ile_akapitów}
    akapitu/ów?"
39  end
40  -%>
41  <%= unless session[ :tylko_nowe_komentarze ]
42  "To są wszystkie akapity z komentarzami."
43  else
44  "To są akapity z nowymi komentarzami"
45  end %>
46  <%= link_to_remote( (unless session[:tylko_nowe_komentarze]
47  '[pokazuj tylko nowe komentarze]'
48  else '[pokazuj wszystkie komentarze]'
49  end ),
50  wsze_lub_nowe_hasz )
51  -%>

54  <% akapity_skomentowane = Akapit.skomentowane(
55  (not session[:tylko_nowe_komentarze]) )
56  -%>

58  <%= render :partial => 'ogladactwo_lista_akapitow', :object =>

```

```
59 [akapity_skomentowane, if Rola.audytork(session[:rola_id]) :  
   "Doradzaj"  
60 else "Superanotuj" end] -%>  
  
62 <%= link_to_remote '[schowaj]', :url => { :action =>  
   :skomentowane_przełącz }  
63 -%>  
  
65 <% else # not session[:skomentowane_pokaż]  
66 -%>  
  
68 <%= link_to_remote '[pokaż]', :url => { :action =>  
   :skomentowane_przełącz }  
69 -%>  
  
71 <%  
72 end # of if session[:skomentowane_pokaż]  
73 end # of if r_wszewid?  
74 -%>
```

### **\_anotuj\_zatwod.rhtml**

```
24 <td>  
25 <%= button_to('zatwierdź',{ :action => 'zatwierdz_zdanie'},  
26 {# :confirm => "Czy rzeczywiście chcesz zatwierdzić zdanie?",  
27 :method => :post}) -%>  
28 </td><td>  
29 <%= button_to('odrzuć',{ :action => 'odrzuć_zdanie'},  
30 {# :confirm => "Czy rzeczywiście chcesz odrzucić zdanie?",  
31 :method => :post}) -%>  
32 </td>  
33 <% if Rola.zarządca?(session[:rola_id]) -%>  
34 <td>  
35 <%= button_to('sądź bliźniacze', { :action => 'zmien_strony_zdan',  
36 :id => @bluzid  
37 },  
38 { :method => :get}) -%>  
39 </td>  
40 <% end -%>
```

### **\_anotuj\_zatwod\_nk.rhtml**

```
24 <% if akat_edytowalny? -%>  
25 <table class="table-plain">  
26 <tr>  
27 <td>  
28 <% pozdost = @akapit_transzy.  
29 poziomy_dostępne(  
30 session[:rola_id],  
31 ## :segmentation_specjalnie => true,  
33 # zmienione 2009/7/28 na życzenie AP  
34 :jak_anotator => session[:jak_anotator],  
35 :poziomy_pokażne => session[:poziomy] )
```

```

36      -%>
37      <table class="table-plain">
38      <tr>
39      <%      pozdost.each do |poz|
40      -%>
41      <td>
42      <%= button_to("&#10003; #{PoziomyAnotacji.kr(poz)}",
43      { :action => 'zatwierdz_akapit', :poziom => poz,
44      :akat => @akapit_transzy },
45      {# :confirm => "Czy rzeczywiście chcesz zatwierdzić akapit?",
46      :method => :post}) -%>
47      </td>
48      <% end # of each |poz|.
49      -%>

51      <% if pozdost.size > 1 -%>
52      <td>
53      <%= button_to(
54      "&#10003; #{pozdost.collect {|poz|
55      PoziomyAnotacji.kr(poz)}.join(', ')}",
56      { :action => 'zatwierdz_akapit',
57      :poziom => pozdost,
58      :akat => @akapit_transzy},
59      {## :confirm => "Czy rzeczywiście chcesz zatwierdzić akapit?",
60      :method => :post}) -%>
61      </td>
62      <% end# of pozdost.size > 1.
63      -%>
64      </tr>
65      </table>
66      </td>
67      <!--
68      -->
69      <% if Rola.zarzadca?(session[:rola_id]) -%>
70      <td>
71      <%=
72      bli = @akapit_transzy.bliźniaczy
73      button_to( 'sądź bliźniaczy',
74      { :action => 'zmien_strony_akapitow',
75      :poziom => :any,
76      :id => ( bli.get_uzid || session[:uzid] ), # ten ostatni dołożony
77      2009/03/05, gdy wyszło, że @bluzid-a może nie być.
78      :akat => bli.id
79      # poziom jest niepotrzebny, bo ta akcja nie przechodzi przez
80      check_edytowalne
81      } ## , { :method => :get }
82      )-%>
83      </td>
84      <% end -%>
85      </tr>

```

```

85   </table>
86   <% end %>

88   <%
89   ## # Local Variables:
90   ## # mode: mmm
91   ## # End:
92   -%>

```

**`_blizniaczy_status.rhtml`**

```

24   <% # wstawiany w widoku ogladactwo
25       abli203 = @akapit_transzy.blizniaczy
26   %>
27   <div id="blizniaczy_status">
28       <%= abli203.status_html %>
29       <% unless r_anotator? # login partnera ma być tajny na
30           # wyraźne życzenie AP 2009/6/26
31       -%>
32       <br/>
33       <%= uz_login( abli203.get_uzid ) %>.
34       <% end %>
35   </div>

37   <%
38   ##       logger.info "@@@ anotuj.rhtml w. 204. #{Time.now}"
40   ## # Local Variables:
41   ## # mode: mmm
42   ## # End:
43   -%>

```

**`_fra_sie_typ.rhtml`**

```

24   <% fraa = fra_sie_typ -%>

26   <fieldset>
27       <% remote_form_for :fraza_sie_typ, @fraza_sie_typ,
28       :url => {:action => :fraza_sie_typ_wybor, :id => fraa,
29       :akat => @akapit_transzy,
30       :poziom => :syntactic, :escape => false} do |form| %>

32       <%=
33       form.select :sie_typ,
34                   FrazaAnot::TYP_SIE,
35       { :prompt => "wyb. typ się",
36       { :onchange =>
37           ajax_request("#{annotation_controller}/fraza_sie_typ_wybor/" +
38           "#{fraa.fraza_anot_id}?poziom=syntactic")}
39       -%>

41       <%= submit_tag "&#10004;" , :class => "submit" -%>

44       <%= link_to_remote '&#10007;',

```

```

45         :url => { :action =>:fraza_wybor_anulowany, :id => fraa, :akat
=> @akapit_transzy, :escape => false}
46     -%>

48 <% end # of form
49 -%>
50 </fieldset>

_fra_wyb_glowy.rhtml

24 <% ktora_g = fra_wyb_glowy[1]
25 if ( ktora_g == :semh or ktora_g == :obie )
26     head_symbol = :semheads
27 else head_symbol = :synheads
28 end
29 fraa = fra_wyb_glowy[0]
30 -%>

32 <fieldset>
33     <% remote_form_for :fraza_glowa, @fraza_glowa,
34 :url => { :action => :fraza_wybor_glowy, :id => fra_wyb_glowy[0],
35 :ktora_g => ktora_g.to_s,
36 :akat => @akapit_transzy,
37 :poziom => :syntactic, :escape => false} do |form| %>

39 <% mbg=fraa.fraz_a_typ.moze_bez_glowy?( head_symbol ) %>
40 <%=
41 form.select :glowa_id,
42     fraa.send( head_symbol, session[:uzid]).collect {|head|
43         [head[2], head[1].token_id]} +
44         if mbg : [[ "(brak" + if mbg==0 : " (!)"; else "" end
+ ")"", -1]] ; else [] end,
45     { :prompt => "wyb. " + ktora_g.to_s },
46     { :onchange =>
47         ajax_request("/#{annotation_controller}/fraz_a_wybor_glowy/" +
48         "#{fraa.fraz_a_anot_id}?ktora_g=#{ktora_g.to_s}" +
49         '&poziom=syntactic') }
50     -%>

52 <%= submit_tag "&#10004;" , :class => "submit" -%>

55 <%= link_to_remote '&#10007;',
56     :url => { :action =>:fraz_a_wybor_anulowany, :id => fraa, :akat
=> @akapit_transzy, :escape => false }
57     -%>

59 <% end # of form
60 -%>
61 </fieldset>

_komentarz_ballot.rhtml

24 <% if r_wszewid?
25     ko = komentarz_ballot

```



```
26     ballot = if ko.nowy? : "&#9744;" else "&#9746" end
27     if r_zarzadca?
28     %>
29     <span id="komentarz_ballot-<%= ko.id %>"><%=
30     link_to_remote( ballot,
31     :url => { :action => :komentarz_nowy_przełącz, :id => ko } ) + ' '
32     %></span>
33     <% else %>
34     <%= ballot + ' ' %>
35     <% end # of if superanotator
36     end # of if wszechwidzący
37     %>
```

### **\_komentarz\_dod\_link.rhtml**

```
25     <div id="komentarz_dodaj">
27     <% unless Rola.gosc?(session[:rola_id]) -%>
28     <%= link_to_remote 'dodaj komentarz',
29     :url => {
30     :action =>:komentarz_dodaj,
31     :id => komentarz_dod_link.akapit_transzy_id} # ,
32     ## :update => 'komentarz_dodaj'
34     -%>
37     <% end # of unless gosc?.
38     -%>
40     </div>
43     <%
44     ## # Local Variables:
45     ## # mode: mmm
46     ## # End:
47     -%>
```

### **\_komentarz\_dodaj.rhtml**

```
24     <% akat, czy_odrzucic = komentarz_dodaj -%>
26     <div id="komentarz_dodaj">
27     <% if session[ :komentarz_dodaj ] -%>
28     <fieldset>
29     <% remote_form_for :komentarz_nowy, @komentarz_nowy,
30     :url => {:action => :komentarz_zatwierdz,
31     :id => akat.akapit_transzy_id,
32     :akapit_id => akat.akapit_id } do |form| %>
34     <% if czy_odrzucic -%>
35     <%= form.select :przyczyna_odrzucenia,
36     Komentarz::PO_select, { :prompt => "wyb. przyczynę"}, { } -%>
37     <br/>
38     <% end # of if czy_odrzucic
39     %>
```

```

40     <%= form.text_area :tresc, :rows => 4, :cols => 60 %>
42     <%= submit_tag "zatwierdź" , :class => "submit" -%>
45     <%= link_to_remote 'anuluj',
46         :url => {
47             :action => :komentarz_anulowany,
48             :id => akat.akapit_transzy_id } # ,
49     ## :update => 'komentarz_dodaj'
51 -%>
53 <% end # of form
54 -%>
55 </fieldset>
57 <% else # not session[:komentarz_dodaj]
58 -%>
59     <% unless Rola.gosc?(session[:rola_id]) -%>
60     <%= link_to_remote 'dodaj komentarz',
61         :url => {
62             :action => :komentarz_dodaj,
63             :id => akat } # ,
64     ## :update => 'komentarz_dodaj'
66 -%>
69 <% end # of unless gosc?.
70 end # of session[:komentarz_dodaj]
71 -%>
73 </div>
75 <%
76 ## # Local Variables:
77 ## # mode: mmm
78 ## # End:
79 -%>

```

**\_komentarze.rhtml**

```

24 <div id="komentarze">
25 <ul>
26 <% # obiekt przekazany tutaj to para akapit transzy, nazwa nadwidoku
27 akat, nadwidok = komentarze
28 if akat.ma_Status?( :<, :do_osadzenia ) and not r_wszewid?
29     uzid = session[ :uzid ] # pokażemy tylko komentarze tego anotatora
30 else uzid = nil # pokażemy wszystkie komentarze
31 end
32     akat.komentarz( uzid, session[
33         "tylko_nowe_komentarze_#{nadwidok}".intern ] ).each do |ko|
34     <li><%= render :partial => 'komentarz_ballot', :object => ko %>
35     <%= ko.created_at.to_s(:db) + ", " +
36     ko.uzytownik.login +
37     "]" + ko.przyczyna.z_przecinkiem( ko.tresc ) -%> </li>

```

```

38  <% end -%>
39  </ul>

41  <% if nadwidok == :anotacja and r_wszewid? %>
42  Teraz obowiązuje ustawienie „pokazuj
43  <%= if session[ :tylko_nowe_komentarze_anotacja ]
44      "tylko nowe"
45  else "wszystkie"
46  end
47  %> komentarze".
48  <%= link_to_remote (if session[ :tylko_nowe_komentarze_anotacja ]
49      '[pokaż wszystkie komentarze]'
50      else '[pokaż tylko nowe komentarze]'
51      end), :url => { :action =>
        :skomentowane_tylko_nowe_anotacja_przełącz,
52      :poziom => :any}
53  -%>
54  <% end %>

56  </div>

59  <%
60  ## # Local Variables:
61  ## # mode: mmm
62  ## # End:
63  -%>

```

## \_lista\_akapitow.rhtml

```

24  <% # Partial wołany z pokaz_transze.rhtml i z wyszukaj_akapity.rhtml.
25  -%>
26  <table>
27  <% for a in lista_akapitow # to jest tablica tablic [akat.akapit, akat].
28      # dawniej było [akat.akapit, akat .status, akat.odrzucony?(,
        akat.uzytownik_id)].
29      a0, a1 = a
30  -%>
31  <tr valign="top" class="<%= cycle('list-line-odd',
32      'list-line-even') %>" >
33  <td><%= a0.akapit_id.to_s +
34      ( u = a1.get_uzid
35      if u != session[ :uzid ]
36      '&nbsp;[' + Uzytkownik.find( u ).login + ']'
37      else "
38      end )-%>
39  </td>
40      <td><%= a0.tresc
41      # to jest tablica [akapit, status] !
42      -%></td>
43      <td>
44      <%= "(#{AkapitTranszy::STATI_kr[ a1.status ]}#{if a1.odrzucony? : '-'}
        end})"

```

```

45  -%>
46  &nbsp;<%= editable = a1.editable_par?( session[ :uzid ] )
47  link_to(
48  if a1.ma_Status?(<:, :zatwierdzony) and editable
49    'Anotuj'
50  elsif editable
51    case a1.status
52    when (AkapitTranszy::STATI[:zatwierdzony])
53      "(czeka na drugiego)"
54    when (AkapitTranszy::STATI[:do_poprawki])
55      "Popraw!"
56    when (AkapitTranszy::STATI[:po_poprawce])
57      "(poprawiony, czeka na drugiego)"
58    when (AkapitTranszy::STATI[:osądzany])
59      "(pod&nbsp;sędem)"
60    when (AkapitTranszy::STATI[:osądzony])
61      "(osądzony)"
62    else 'Oglądaj'
63    end # of case.
64  else 'Oglądaj'
65  end + # of if
66    if pokr = PoziomyAnotacji.
67      krótkie( a1.poziomy_annotacje( session[:rola_id] ) ) and
68      pokr[0]
69      ' <b>' + pokr.join(', ') + '</b>'
70    else "
71    end,
72    {:action => 'ustal_akapit',
73     ##           :transza => @transza,
74     :akapit => a0,
75     :akat => a1,
76     :status => a1.status,
77     :anotator => a1.uzytkownik_id} # to będzie nil gdy z widoku
78     pokaz_transze, i dobrze.
79  ) -%></td>
80  </tr>
81  <% end -%>
82  </table>

84  <%
85  ## # Local Variables:
86  ## # mode: mmm
87  ## # End:
88  -%>

```

### \_lista\_odebranych.rhtml

```

24  <% if session[:odebrania_transz_pokaz] and lista_odebranych and
25  lista_odebranych[0] -%>
    <h3> Transze odbierane i odebrane </h3>

```

```

27  <ul>
28  <% lista_odebranych.each do |odtra| -%>
29  <li>
30  <%= odtra.dotyczy.opis + " [" +odtra.created_at.to_s(:db)+"]" -%>

32  <b><% if odtra.spelniona? -%>
33    odebrana <%= odtra.updated_at.to_s(:db) %>
34  <% else # nie odebrana
35  -%>
36  oczekuje na odebranie
37  <% end # of if odebrana?
38  -%>
39  </b>
40  <br/>
41  <%= odtra.opis -%>
42  <br/>
43  </li>
44  <% end # of each odebranie transzy
45  -%>
46  </ul>
47      <%= link_to_remote '[ukryj listę odebrań]',
48      :url => {:action => 'odebrania_transz_ukryj',
49      :id => lista_odebranych[0].uzytkownik_id},
50      :update => 'lista_odebranych' %>
51      <% elsif lista_odebranych and lista_odebranych[0] # nie
52      pokazuj odebrań
53      -%>
54      <%= link_to_remote '[pokaż listę odebrań transz]',
55      :url => {:action => 'odebrania_transz_pokaz',
56      :id => lista_odebranych.uzytkownik_id},
57      :update => 'lista_odebranych' %>
58  <% end # of if session[:odebrania_transz_pokaz]
59  -%>

```

## \_lista\_transz.rhtml

```

24  <!-- woła nas widok lista_transz i ogladactwo-->

27  <ul>
28  <% lista_transz.each do |tra| %>
29  <li>
30  <%= tra.opis %>
31  <% if r_anotator? -%>
32  <%= dopo = tra.do_poprawki?
33      if dopo : "<b>#{dopo} do poprawki!</b>" end
34  %>
35  <% elsif r_zarzadca? %>
36  <%= saro = tra.sa_rozbiezne?
37      if saro : "<b>rozb. w #{saro}</b>" end
38  %>
39  <% end -%>

```

```

40      <%= link_to "[przejdź]" , { # link_to options
41        :controller => annotation_controller,
42        :action => 'pokaz_transze',
43        :id => tra
44      },
45      { # html options
46        :method => :get
47      }
48    %>

50  </li>
51  <% end %>
52 </ul>

54 <% if lista_transz.size < Transza::MAX_TRANSZ and r_anotator? %>
55 <%= link_to_remote "[pobierz transzę]" ,
56   :url => { # link_to options
57     :controller => annotation_controller, :action =>
58     'pobierz_transze', :id => @anotator, :escape => false

60   }, :method => :post, :update => 'lista_transz'
61 %>
62 <% end -%>

```

### \_nowa\_segmentacja.rhtml

```

24 <% url_h = { :action => :nowa_prośba_segmentacyjna }
25 if akid183 = nowa_segmentacja
26 url_h.update( :akapit => akid183 )
27 end
28 url_h.update( :akat => session[ :akat ] ) if session[ :akat ]
29 remote_form_for ( :nowa_segmentacja, nil,
30 :url => url_h ) do |form| %>
31   <p>
32     <label for="is" >Orth(y), który/e chcesz podzielić/połączyć
33     (jeśli połączyć – napisz rozdzielone spacjami) </br> <span
34       class="sc">Uwaga!</span>
35     Kasztoczułe (case sensitive)!</label>
36     <br/>
37     <%= form.text_field :is, :size => 40 %>
38   </p>
39   <p>
40     <label for="should_be" >proponowany podział/połączenie (jeśli
41     podział
42     – rozdziel spacjami)
43     <br/> <span class="sc">Uwaga!</span>
44     Kasztowość (upper/lower case) musi się zgadzać z tym, co wyżej!!
45     <br/>
46     <%= form.text_field :should_be, :size => 40 %>
47   </p>

```

```

48     <% [ true, false ].each {|decision| -%>
49     <%= form.radio_button :nps, decision %>
50     <%= if decision : "docelowo z brakiem spacji"
51     else "docelowo rozdzielone spacją(ami)"
52     end + " &nbsp;&nbsp;&nbsp;" -%>
53     <% } # of each decision
54     -%>

56     <% if r_wszewid? %>
57     <p>
58         <label for="akids" >id(y) akapitu(ów) (gdy więcej niż jeden -
59         rozdziel spacjami)</label>
60         <br/>
61         <%= form.text_field :akids, :size => 25, :value => ( akid183 ||
        "wszystkie_pasujące" ) %>
62     </p>
63     <% end # of if wszewid
64     -%>
65     <p>
66         <label for="opis" >opis (opcjonalny) </label>
67         <br/>
68         <%= form.text_area :opis, :rows => 4, :cols => 40 %>
69     </p>
70     <%= submit_tag "Wygeneruj prośbę" , :class => "submit" -%>
71     <% end # of do |form|
72     -%>

```

## \_odbierz\_transze.rhtml

```

24     <% anotator = odbierz_transze
25     transze_select = anotator.transze_otwarte.collect{ |tra|
26         [tra.opis_nohtml, tra.transza_auto_id] }
27     -%>
28     <fieldset>
29         <% remote_form_for :prosba_o_odebranie, @prosba,
30         :url => {:action => :prosbe_o_odebranie_zapisz,
        :id => anotator } do |form| %>
33         <%= form.select :transza_auto_id,
34         transze_select, {:prompt => "wyb. transzę"}, { } -%>
35     <br/>
36     <legend>Wpisz przyczynę prośby</legend>
37     <%= form.text_area :przyczyna, :rows => 4, :cols => 60 %>
39     <%= submit_tag "zatwierdź" , :class => "submit" -%>
42     <%= link_to_remote 'anuluj',
43         :url => {:action => :prosba_anulowana, :id => anotator },
44         :update => 'odbierz_transze'
45     -%>
47     <% end # of form
48     -%>

```

49       </fieldset>

### **\_ogladactwo\_goto.rhtml**

```

25     <fieldset>
26       <% form_for :goto, :url => { :action => :ustal_akapit } do |form|
27        %>
28        <p>
29          <label for="akapit">akapitu nr:</label>
30          <%= form.text_field :akapit, :size => 10 -%>
31          <%= submit_tag "przejdź" , :class => "submit" %>
32        <br/>
33        <%= form.text_field :transzy, :size => 10 -%>
34        <%= submit_tag "przejdź" , :class => "submit" %>
35        <br/>
36        <label for="akat">akapitu transzy nr:</label>
37        <%= form.text_field :akat, :size => 10 -%>
38        <%= submit_tag "przejdź" , :class => "submit" %>
39        <br/>
40        (wystarczy podać jedno, akapit transzy przeważa nad akapitem;
41        znaki niebędące cyframi dziesiętnymi są ignorowane) </p>
42        <%= end # of form.
43        -%>
44     </fieldset>
45     <%
46        ## # Local Variables:
47        ## # mode: mmm
48        ## # End:
49        -%>

```

### **\_ogladactwo\_lista\_akapitow.rhtml**

```

24     <% tab_akapitow, akcja, no_inf = ogladactwo_lista_akapitow
25     -%>
26     <table>
27       <% tab_akapitow.each do |aka|
28        akat = aka.akat0
29        anotid = akat.get_uzid
30        -%>
31        <tr valign="top" class="<%= cycle('list-line-odd',
32        'list-line-even') %>" >
33        <td><%= h aka.akapit_id %>.
34        <% if r_wszewid? %>
35        (<%= h aka.akat0.id %>)
36        <% end -%>
37        </td>
38        <td><%= aka.tresc -%>
39        <% if aka.respond_to?( "kcrea" )
40        kcrea = aka.kcrea
41        -%>

```



```
42     render :partial => 'komentarze', :object => [ akat, :ogladactwo ]
43   end %>

44   </td>
45   <td>
46     <%= link_to( akcja.to_s,
47       :action => 'ustal_akapit',
48       ##           :transza => @transza,
49       :akapit => aka.akapit_id ,
50       :akat => akat,
51       :anotator => anotid # dla ustalenia akapitu transzy redundantny,
52       # ale służy do zaznaczenia, że prawnym
53       # anotatorem akapitu jest ktoś inny od osoby, która teraz anotuje.
54       )
55     -%></td>
56   </tr>
57   <% end # of each akapit
58   -%>
59 </table>
60
61 <%
62 ## # Local Variables:
63 ## # mode: mmm
64 ## # End:
65 -%>
```

## **\_prosba\_link.rhtml**

```
24   <%= link_to_remote '[prośba o odebranie transzy]',
25     :url => { :action => 'prosba_o_odebranie_transzy', :id =>
26       prosba_link,
27       :update => 'odbierz_transze' %>
```

## **\_prosby\_anotatorek.rhtml**

```
24   <% # partial wołany z widoku ogladactwo (dostępnego wyłącznie dla
25     # bądź anotuj.
26   if r_wszewid? -%>
27   <h1>Prośby Anotatorów</h1>
28   <% end -%>

31   <% if nseg = (flash[ :nowa_segmentacja ] || flash[ :prosby_anotatorek
32     ]) -%>
33   <div id="notice">
34     <%= nseg -%>
35   </div>
36   <% end -%>

38   <% if r_wszewid?
39     nowe_prosby = ProsbyAnotatorek.nowe( session[ :rola_id] )
40   if nowe_prosby and nowe_prosby[0] -%>
41   <ul>
```

```

42 <% nowe_prosby.each do |prosba| -%>
44 <li><%= prosba.uzytkownik.login # + ", "+ prosba.rodzaj
45 -%>
47 (id prośby: <%= prosba.id %>)
49 <%= link_to_remote '[spełnij prośbę]',
50 :url => { :action => :spełnij_prośbę ,
51 :id => prosba.id,
52 :dotyczy_id => prosba.dotyczy_id,
53 :dotyczy_type => prosba.dotyczy_type, :escape => false },
54 :confirm => prosba.konfirmacja
55 -%>
57 <%= link_to_remote '[odrzuć prośbę]',
58 :url => { :action => :odrzuć_prośbę ,
59 :id => prosba.id },
60 :confirm => "Czy rzeczywiście chcesz odrzucić tę prośbę?"
61 -%>
62 <br/> <%= prosba.treść_z_opisem %>
64 <%=
65 if (dotycz = prosba.dotyczy).is_a?( NowaSegmentacja )
66 aids = dotycz.ids
67 text = "
68 text = "<p>#{aids.size} akapit/y/ów, począwszy od:</p>" if
aids.size > 5
69 aids[0, 5].compact.each{ |aid| text += "<p><b>Akapit #{aid}</b> " +
70 "#{Akapit.find( aid ).tresc}</p>" }
71 text
72 end
73 -%>
75 </li>
77 <% end # of nowe_prosby.each |prosba|
78 -%>
79 </ul>
81 <% end # else not nowe_prosby
82 -%>
83 <% if r_zarzadca? and session[:odebrania_bp_pokaz] -%>
85 <% remote_form_for ( :transza_odbierana, nil,
86 :url => { :action => :odbierz_transze_bprosby } ) do |form| %>
87 <p>
88 <label for="transza_odbierana">Numer transzy, którą chcesz
odebrać</label>
89 <br/>
90 <%= form.text_field :numer, :size => 4 %>
91 </p>
92 <%= submit_tag "Wygeneruj prośbę" , :class => "submit" -%>
93 <% end # of do |form|

```

```

94     -%>

96     <%= link_to_remote '[ukryj generator prośby]',
97         :url => { :action => 'odebrania_bp_ukryj' },
98     :update => 'prosby_annotorek' -%>

100     <% elsif r_zarzacca? # zarzacca, ale nie pokazać odebrań bp
101     -%>

103     <%= link_to_remote '[wygeneruj prośbę o odebranie transzy]',
104         :url => { :action => 'odebrania_bp_pokaz' },
105     :update => 'prosby_annotorek' -%>

107     <% end # of if zarzacca and pokazać odebrania bp
108     if session[ :nowa_segmentacja_pokaz ] -%>

110     <%= render :partial => 'nowa_segmentacja', :object => nil -%>

112     <%= link_to_remote '[ukryj generator prośby]',
113         :url => { :action => 'nowa_segmentacja_ukryj' },
114     :update => 'prosby_annotorek' -%>

117     <% end # of if pokazuj segmentacje
118     -%>

120     <%= link_to_remote '[wygeneruj prośbę o nową segmentację]',
121         :url => { :action => 'nowa_segmentacja_pokaz' },
122     :update => 'prosby_annotorek'
123     -%>

126     <% # end of if nowe_prosby (else)
127     else # not r_wszewid?
128     -%>

129     <% if akat_proszalny_segmentnie?
130     if session[ :nowa_segmentacja_pokaz ] -%>
131     <%= render :partial => 'nowa_segmentacja', :object =>
132     @akapit_transzy.akapit_id -%>

134     <%= link_to_remote '[ukryj zgłaszacz prośby]',
135         :url => { :action => 'nowa_segmentacja_ukryj', :akat =>
            @akapit_transzy },
136     :update => 'prosby_annotorek' -%>

138     <% else -%>
139     <%= link_to_remote '[zgłoś prośbę o nową segmentację]',
140         :url => { :action => 'nowa_segmentacja_pokaz', :akat =>
            @akapit_transzy },
141     :update => 'prosby_annotorek'
142     -%>

144     <% end # of session[:nowa_segmentacja_pokaz]
145     end # of akat edytowalny (przez Anotatora)
146     end # of if r_zarzacca? or not
147     -%>

150     <%

```

```

151  ## # Local Variables:
152  ## # mode: mmm
153  ## # End:
154  -%>

```

### **\_przyklady\_sensow\_link.rhtml**

```

24  <% sens = przyklady_sensow_link -%>

```

### **\_quasi\_transza.rhtml**

```

24  <% klucz = quasi_transza[1]
25  qt = quasi_transza[0] -%>

28  <% if ( not session[ :pokaz_qtra ] ) or not session[ :pokaz_qtra ][
    klucz ] -%>
29  <%= link_to_remote '[' + klucz + ']' ('+ qt.size.to_s + '
    akapit/y/ów)',
30      :url => {
31          :action =>:pokaz_quasi_transze,
32          :klucz => klucz},
33      :update => 'quasi_transza_' + klucz
34  -%>

36  <%= link_to '[' + klucz + ']' (na osobnej stronie)',
37      :action =>:quasi_transza_bruta1,
38      :klucz => klucz
39  -%>

41  <% else # of session[ :pokaz_qtra...].
42  -%>
43  <%= link_to_remote "[schowaj #{klucz}]",
44      :url => {
45          :action =>:ukryj_quasi_transze,
46          :klucz => klucz},
47      :update => 'quasi_transza_' + klucz
48  -%>
49  <%= render :partial => 'ogladactwo_lista_akapitow', :object =>[qt,
    :Oglądaj, :no_inf] -%>
50  <% end # of session[ :pokaz_qtra...].
51  -%>

54  <%
55  ## # Local Variables:
56  ## # mode: mmm
57  ## # End:
58  -%>

```

### **\_sens\_long.rhtml**

```

24  <%
25  akatid191, tokid191, sens191 = sens_long
26  eltid191 = "long-sense_#{akatid191}_#{tokid191}_#{sens191.id}"

```

```
27 url_hash191= {:url => {
28   :action => :sense_long_toggle, :id => eltid191, :akat => akatid191,
29   :escape => false, :poziom => :word_senses } }

31 sls191 = session[:show_long_sense]
32 if sls191 and sls191.include?( eltid191 )
33   -%>
34   <div id="%= eltid191 %" style="border: 1px">
35     <%= sens191.long_description -%>
36     <%= link_to_remote 'ukryj pełny opis', url_hash191 -%>
37   </div>

39   <% else # not to show long description
40     -%>

42   <span id="%= eltid191 %"></span>

44   <% end # of show or don't show long descr.
45   -%>

47   <%
48   ## # Local Variables:
49   ## # mode: mmm
50   ## # End:
51   -%>
```

**\_sens\_pelny\_opis\_link.rhtml**

```
24 <%
25 akatid191, tokid191, sens191 = sens_pelny_opis_link
26 eltid191 = "long-sense_#{akatid191}_#{tokid191}_#{sens191.id}"
27 url_hash191= {:url => {
28   :action => :sense_long_toggle, :id => eltid191, :akat => akatid191,
29   :escape => false, :poziom => :word_senses } }

32 sls191 = session[:show_long_sense]
33 if sls191 and sls191.include?( eltid191 )
34   -%>
35   <span id="%= eltid191 %"-link">
36     <%= link_to_remote 'ukryj pełny opis', url_hash191 -%>
37   </span>

39   <% else # not to show long description
40     -%>

42   <span id="%= eltid191 %"-link">
43     <%= link_to_remote 'pełny opis', url_hash191 -%>
44   </span>

46   <% end # of show or don't show long descr.
47   -%>

49   <%
50   ## # Local Variables:
51   ## # mode: mmm
```

```

52  ## # End:
53  -%>

```

### **\_sentenśnik.rhtml**

```

24  <%
25      akat = sentenśnik
26      akat.akapit.token.each do |token|
27          kończy_zdanie = KoniecZdaniaAnot.znajdz( akat, token.id )
28      -%>
29      <%= link_to_remote (
30          '&nbsp;' +
31          token.orth + (
32              if kończy_zdanie : ' &nbsp;';
33              else '&nbsp;';
34              end ) ) ,
35          { :url => {
36              :action => :konczymy_zdanie,
37              :id => token.id,
38              :akat => akat,
39              :poziom => :sentences, :escape => false }
40          }, { :class => "bright" }
41      %><%= if token.ns_nastepuje? : '&#8239;';
42          else '&#8196;&#8196;'; # em/3; &#8195 em space; &#8194 en space
          &#32; są też three-per-em, four-per-em, six-per-em spaces.
43      end -%>
44      <% end # of each token
45      -%>

```

### **\_status\_info.rhtml**

```

24  <%= @akapit_transzy.status_html +
25      if DlaEli.nk : "<br/>" else " " end +
26      "(#{uz_login( @anotid )})" +
27      if atua = @akapit_transzy.updated_at
28          ", " + atua.to_s(:db)
29      else ""
30      end + ")" +
31      if ( ( not r_anotator?) or session[ :drag_queen_id ] or UNatrora.x )
32      and @bluzid :
33          " (#{uz_login( @bluzid )},
          tr.&#8201;#{@akapit_transzy.bliźniaczy.transza_id},
          at.&#8201;#{@akapit_transzy.bliźniaczy_id})" end.to_s
33  %>

```

### **\_token\_disamb.rhtml**

```

24  <!-- wywołujemy go dla obiektu token klasy Token z widoku anotuj.rhtml -->
25  <% token, akat, lewy = token_disamb
26  akath = akat.akat_hash
27  akat_e = akat_edytowalny?( :morphosyntactic, akat, lewy ) -%>

```

```

29   <b>
30   <% tdisambs = token.disambs( akath )
31   -%>
32   <% interps = ( if tdisambs[ 0 ] : tdisambs
33       else token.interpretacja
34       end )
35   interps.each {|interp| -%>
36   <span class="sc"><%= interp.leksem.lemat %></span>
37   <%= interp.fulltag -%>
38   <% if interps.index( interp ) + 1< interps.size -%>
39   <br>
40   <% end -%>
41   <% } -%>
42   </b>

44   <% if akat_e and
45       ((not token.interp?( akath )) or
46       (not (di = token.the_disambiguation )) or
47       di.dodana? )
48   -%>
49   <%= link_to_remote ( if tdisambs.size != 1 : 'dezambiguuj!' else
50       'wybierz' end),
51       :url => {
52         :action =>:pokaz_interpretacje,
53         :id => token.id,
54         :akat => akat,
55         :poziom => :morphosyntactic, :escape => false },
56   :update => "disa-token_{akat.id}_{token.id}" -%>

58   <%= link_to_remote 'dodaj',
59       :url => {
60         :action =>:token_tworz_tag,
61         :id => token.id,
62         :akat => akat,
63         :poziom => :morphosyntactic, :escape => false },
64   :update => "disa-token_{akat.id}_{token.id}" -%>

66   <%= if r_zarzadca?
67       link_to_remote znak_zatwierdź_tę,
68       :url => {
69         :action => :morfoskładnia_zatwierdź_tę,
70         :id => token.id,
71         :akat => akat,
72         :disamb_id => tdisambs[0].id,
73         :poziom => :morphosyntactic, :escape => false }
74   end -%>

76   <% end # of akat_e and not token.interp? etc.
77   -%>

79   <%
80   ## # Local Variables:

```

```

81  ## # mode: mmm
82  ## # End:
83  -%>

```

### **\_token\_fra.rhtml**

```

24  <% token = token_fra[0]
25  akat_hash = token_fra[1]
26  tokid = token.token_id
27  lewa = token_fra[2] # dla wywołania po lewej stronie ma być not nil a po
    prawej nil
28  akat_e = akat_edytowalny?( :syntactic )
29  ## logger.info "@@@ akat_e: #{akat_e}"
31  -%>

34  <% if lewa : akat = @akapit_transzy # zainicjowane np. przez akat_hash
    w kontrolerze
35  else akat = @akapit_transzy.bliźniaczy
36  end
37  -%>

41  <% fraza_gdy_reszta = akat.reszta_frazy?(tokid)
42  fra = akat.pocz_frazy?(tokid)
43  # jeśli któraś z tych metod dopasuje tokid, to zwraca odpowiednią
44  # frazę_anot
45  if fraza_gdy_reszta -%>

47  <!--
    -->

49  <% elsif fra -%>
50  <!-- renderujemy -->

52  <td class="<%=
53      if @weryfikacja and @weryfikacja[:frazy] and
54      @weryfikacja[:frazy].include?(tokid)
55          'phrase-cell-alert'
56  else
57      case akat.indeks_frazy(tokid) % 2
58      when 0: 'phrase-cell-odd'
59      when 1: 'phrase-cell-even'
60      end
61  end
62  %>" id="token_fra-<%=if lewa : tokid; else -tokid end %>"
    rowspan="<%=fra.dlugosc %>">

64  <% ft = fra.fraza_typ
65  if ft and (not @pokaz_wybor_typu) -%>
66  <%= ft.typ_symbol -%>

68  <% if akat_e and lewa -%>
69  <%= link_to_remote 'zmień',
70      :url => { :action => :frazuukryj_typ ,

```



```

71      :id => fra, :poziom => :syntactic, :akat => @akapit_transzy, :escape
      => false },
72      :update =>
      "token_fra_#{akat_hash[:akapit_transzy_id]}_#{tokid}"
73      -%>
74      <% end -%>
75      <br>
77      <% if ft.typ_symbol == "VP" -%>
78      <%= if fra.vp_typ == 1: "pos,"
79      elsif fra.vp_typ == 0: "neg,"; end -%>
80      <% end -%>
82      <% if ft.raczej_jednoglowa? and
83      fra.token_id_synhead==fra.token_id_semhead and
84      not @pokaz_dwie_glowy
85      %>
86      <% if fra.token_id_semhead and (not @pokaz_wybor_semh)
87      fratsemh = fra.token_semhead
88      if fratsemh
89      -%>
90      head: <%= fratsemh.lemat( akat_hash ) -%>
91      <% end # of if fratsemh
92      -%>
93      <% if akat_e and lewa and
94      fra.semheads( akat_hash ).size >1
95      -%>
96      <%= link_to_remote 'zmień', :url => {
97      :action => :fraza_glowy_ukryj, :id => fra,
98      :poziom => :syntactic, :akat => @akapit_transzy, :escape => false }
99      -%>
100     <% end # of if edytowalny
101     -%>
102     <% if akat_e and lewa and
103     (not ft.jednoglowa?) and
104     ( fra.semheads( akat_hash ) | fra.synheads( akat_hash ) ).size >1
105     -%>
106     <%= link_to_remote '2gł.', :url => {
107     :action => :fraza_dwie_glowy, :id => fra,
108     :poziom => :syntactic, :akat => @akapit_transzy, :escape => false }
109     -%>
110     <% end # of fra.token_id_semhead and (not @pokaz_wybor_semh)
111     -%>
112     <% elsif ( not fra.token_id_semhead ) and
113     (not @pokaz_wybor_semh) and
114     ft.bezglowa?
115     -%>
116     (brak head)
117     <% if akat_e and lewa and
118     fra.semheads( akat_hash ).size >1

```

```

119     -%>
120     <%= link_to_remote 'zmień', :url => {
121         :action => :frazaglowyukryj, :id => fra,
122         :akat => @akapit_transzy,
123         :poziom => :syntactic, :escape => false } -%>
124     <% end -%>

126     <% else # fraza nie ma głowy a musi mieć lub chcemy pokazać wybór
        głowy
127     -%>
128     <%= render :partial => 'fra_wyb_glowy',
129     :object => [fra, :obie] -%>

131     <% end -%>

133     <% else # fraza dwugłowa
134     %>
135     <% if fra.token_id_synhead and (not @pokaz_wybor_synh)
136     fratsynh = fra.token_synhead
137     if fratsynh
138     -%>
139     synh.: <%= fratsynh.lemat( akat_hash ) -%>
140     <% end # of if fratsynh
141     -%>
142     <% if akat_e and lewa and
143     fra.synheads( akat_hash ).size >1
144     -%>
145     <%= link_to_remote 'zmień', :url => {
146         :action => :frazasynhukryj, :id => fra,
147         :akat=> @akapit_transzy,
148         :poziom => :syntactic, :escape => false } -%>
149     <% end -%>

151     <% elsif ( not @pokaz_wybor_synh ) and
152     ( not fra.token_id_synhead ) and
153     ft.moze_bez_glowy?( :synh )
154     %>
155     (brak synh.)
156     <% if akat_e and lewa and
157     fra.synheads( akat_hash ).size > 0
158     -%>
159     <%= link_to_remote 'zmień', :url => {
160         :action => :frazasynhukryj, :id => fra,
161         :akat => @akapit_transzy,
162         :poziom => :syntactic, :escape => false} -%>
163     <% end -%>

165     <% else # fraza nie ma synh lub chcemy ją ukryć żeby zmienić wybór
166     -%>
167     <%= render :partial => 'fra_wyb_glowy',
168     :object => [fra, :synh] -%>
169     <% end -%>

```

```

171    <br>

173    <% if fra.token_id_semhead and (not @pokaz_wybor_semh)
174        fratsemh = fra.token_semhead
175        if fratsemh
176            -%>
177        semh.: <%= fratsemh.lemat( akat_hash ) -%>
178        <% end # of if fratsemh
179            -%>
180        <% if akat_e and lewa and
181            fra.semheads(akat_hash).size >1 -%>
182            <%= link_to_remote 'zmień', :url => {
183                :action => :fraza_semh_ukryj, :id => fra,
184                :akat => @akapit_transzy,
185                :poziom => :syntactic, :escape => false} -%>
186            <% end -%>

188        <% elsif ( not @pokaz_wybor_semh ) and
189            ( not fra.token_id_semhead ) and
190            ft.moze_bez_glowy?( :semh )
191            %>
192            (brak semh.)
193        <% if akat_e and lewa and
194            fra.semheads(akat_hash).size > 0
195            -%>
196            <%= link_to_remote 'zmień', :url => {
197                :action => :fraza_semh_ukryj, :id => fra,
198                :akat => @akapit_transzy,
199                :poziom => :syntactic, :escape => false} -%>
200            <% end -%>

203        <% else # fraza nie ma głowy sem. lub chcemy ją ukryć żeby zmienić
                wybór
204            -%>
205            <%= render :partial => 'fra_wyb_glowy',
206                :object => [fra, :semh] -%>
207            <% end # of fraza nie ma głowy sem./ma
208            -%>

210        <% end # of jednogłowa/dwugłowa
211            -%>

213        <% if ft.typ_symbol == "sie" and DlaEli.x -%>
214        <% if fra.sie_typ and (not @pokaz_wybor_sie_typu) -%>
215        <!-- &#0132;się&#0148;-->
216        <br>
217        <%= FrazaAnot::TYP_SIE[fra.sie_typ][0] %>
218        <% if akat_e and lewa -%>
219        <%= link_to_remote 'zmień', :url => {
220            :action => :fraza_sie_typ_ukryj, :id => fra,
221            :akat => @akapit_transzy,
222            :poziom => :syntactic, :escape => false} -%>

```

```

223 <% end -%>
224 <% else # sie_typ is nil
225 -%>
226 <%= render :partial => 'fra_sie_typ',
227 :object => fra %>

229 <% end -%>
230 <% end # of typ_symbol == sie
231 -%>

233 <% else # fraza dotąd nie ma
234 # ustalonego typu lub chcemy go być może zmienić
235 -%>

237 <fieldset>
238 <% remote_form_for :fraz_a_typ, @fraz_a_typ,
239 :url => { :action => :fraz_a_wybor_typ_u, :id => fra,
240 :ak_a => @akapit_transzy,
241 :poziom => :syntactic, :escape => false } do |form| %>

243 <%=
244 form.select :fraz_a_typ_id,
245 Fraz_aTyp.wybor,
246 { :prompt => "wyb. typ" },
247 { :onchange =>
248 ajax_request("/#{annotation_controller}/fraz_a_wybor_typ_u/" +
249 "#{fra.fraz_a_anot_id}?poziom=syntactic") }
250 -%>

251 <%= submit_tag "&#10004;" , :class => "submit" -%>

254 <%= link_to_remote '&#10007;',
255 :url => {
256 :action => :fraz_a_wybor_anulowany, :id => fra, :ak_a =>
257 @akapit_transzy }
258 -%>

259 <% end # of form
260 -%>

261 </fieldset>

263 <% end # of if ft
264 -%>

267 <% if akat_e and lewa -%>
268 <%= # button, bo musimy zrenderować całą stronę, żeby odbudować
269 # brakujące komórki
270 button_to( 'wyczyść frazę', {
271 :action => :fraz_a_wyczysc,
272 :id => fra.fraz_a_anot_id,
273 :ak_a => @akapit_transzy,
274 :poziom => :syntactic },
275 {# :confirm => "Czy rzeczywiście chcesz wyczyścić tę frazę?",
276 :method => :post})

```

```

277     -%>
278     <% end -%>

281     </td>
282     <% else # nie początek ani środek frazy
283     -%>
284     <td class="<%=
285         if @weryfikacja and @weryfikacja[:frazy] and
286         @weryfikacja[:frazy].include?(tokid)
287         'alert-bg'
288     end %>" id="token_fra_<%=if lewa : tokid; else -tokid end %>">

290     <% if akat_e and lewa and
291     (not token.interp?(akat_hash))
292     -%>
293     <div class="chwytalny" id=<%= "chwyc_#{if lewa : tokid; else -tokid
294     end}" %>>
295         chwyc
296     </div>
297     <%= draggable_element( "chwyc_#{if lewa : tokid; else -tokid end}",
298     :revert => true) -%>
299     <div id=<%= "upusc_#{if lewa : tokid; else -tokid end}" %>>
300         upuść
301     </div>
302     <%= drop_receiving_element(
303         "upusc_#{tokid}",
304         :url =>
305         { :action => 'frazu_upuszczono', :id => tokid, :akat =>
306         @akapit_transzy },
307         :accept => "chwytalny",
308         :dropempty=> "true",
309         :hoverclass => 'hover',
310         :loading => visual_effect(:fade),
311         :with => "'upusc=' +
312         encodeURIComponent(element.id.split('_').last())" ,
313         :complete => visual_effect(:highlight, "token_fra_#{if lewa :
314         tokid; else -tokid end}")
315     )
316     -%>
317     <% end # of if akat_e<dytowany(:syntactic)>.
318     -%>

319     </td>
320     <% end # of if wefrazie? etc.
321     -%>

322     <%
323     ## # Local Variables:
324     ## # mode: mmm
325     ## # End:
326     -%>

```

**\_token\_framka.rhtml**

```

24  <% tokid = token_framka[0].token_id
25      at_hash = token_framka[1]
26      at = @akapit_transzy # zainicjowane np. przez akat_hash kontrolera
27      wefra = at.we_frazie?( tokid )
28      -%>
29  <td class="phrase-<%=
30      case wefra
31      when 0 : 'mid'
32      when 1 : 'beg'
33      when 2 : 'end'
34      when 3 : 'whole'
35      end
36      %>" id="framka_<%= tokid %>">
37      &nbsp;  </td>

```

**\_token\_segm.rhtml**

```

24  <% # partial renderowany przez anotuj.rhtml, a z niego
25      # _token_td.rhtml,
26      # tylko dla tokenów,
27      # które mają przypisany id wyboru segmentacji. Jesteśmy pewni, że
28      # renderujemy go tylko dla tokenu, który mamy prawo edytować, czyli
29      # w szczególności zawsze dla tokenu po lewej stronie.
30      akat, token, sgvan = token_segm
31      zmieniaj = session[ :zmieniaj_segmentacje ]
32      zmieniaj = zmieniaj[ token.id ] if zmieniaj
33      unless zmieniaj -%>
34      <%= if token.interp?( akat.akat_hash )
35          token.orth
36      end %>
37      <br/><%= if sgvan
38          if sgvan.chosen? : "(ta segm. wybrana)"
39          else "(ta segm. odrzucona)"
40          end
41      end
42      -%>
43      <%= link_to_remote (if sgvan : 'zmień' else 'zdecyduj' end), :url =>
44      { :action => 'segmentacja_zmieniaj',
45        :id => token.id, :akat => akat, :poziom => :segmentation, :escape =>
46        false } -%>
47
48      <%= if r_zarzadca? and
49          ( sgvan.chosen? or token.choice_card <= 2 ) # tylko gdy jesteśmy
50          # w wybranym lub mamy co najw. 2 możliwości.
51          link_to_remote znak_zatwierdź_tę, :url => { :action =>
52              'segmentacja_zatwierdź_tę',

```

```

51      :id => token.id, :akata => akat, :poziom => :segmentation, :escape =>
false }
52    end -%>

55    <% else # czyli if zmieniaj
56    -%>

57    <fieldset>
58      <% remote_form_for :sg_variant_anot, @sg_variant_anot,
59      :url => { :action => :segmentacja_ujednoznacznij,
60        :id => token.id ,
61        :akata => akat,
62        :poziom => :segmentation, :escape => false} do |form| -%>

65      <% token.sg_possies.each {|decision| -%>
66        <%= form.radio_button :wybierz, decision,
67        {:onclick =>
68          ajax_request("/#{annotation_controller}/segmentacja_ujednoznacznij/#{token.id}?poziom=segm
69          %>

71      <%= if decision : "wybierz tę segm." else "odrzuć tę segm." end + "
&nbsp;&nbsp; " -%>
72      <% } # of each decision
73      -%>

75      <%= ## submit_tag "&#10003;" , :class => "submit"
77      # schowany 2009/03/06
78      -%>

80      <%= link_to_remote '&#10007;', :url => { :action =>
81      :segmentacja anulowana, :id => token.id, :akata => akat, :escape =>
82      false }
83      -%>

85    <% end # of form
86    -%>

87    </fieldset>

90    <% end # of unless zmieniaj.
91    -%>

94    <%
95    ## # Local Variables:
96    ## # mode: mmm
97    ## # End:
98    -%>
```

token sens.rhtml

```

24  <% token, akat, lewy = token_sens
25  if disamb_id = token.do_sensu?( akat.akat_hash )
26    @the_sens = token.the_sens( akat.akat_hash ) # zauważmy, że zmienna
           ta jest niby klasowa, ale tak naprawdę jej wartość jest ustalana osobno w
           każdej instancji tego partiala, a zatem dla wszystkich wybs-ów zstępnych.
27  -%>

```

```

29  <%      if @the_sens  # radio button potrzebuje zmiennej instancyjnej
30  -%>

32  <%= @the_sens.short_description  %>
33  <%= render :partial => 'sens_pełny_opis_link', :object => [akat.id,
    token.id, @the_sens]
34  -%>
35  <%= render :partial => 'sens_long', :object => [akat.id, token.id,
    @the_sens]
36  -%>

38  <% if akat_edytowalny?( :word_senses, akat, lewy ) %>
39  <%= link_to_remote 'zmień',
40      :url => {
41          :action =>:pokaz_sensy,
42          :id => token.id,
43          :akat => akat,
44          :poziom => :word_senses, :escape => false},
45          :update => "sens-anot_#{akat.id}_#{token.id}"
46  %>
47  <%= if r_zarzacca?
48      link_to_remote znak_zatwierdź_tę,
49      :url => {
50          :action => :word_senses_zatwierdź_tę,
51          :id => token.id,
52          :akat => akat,
53          :poziom => :word_senses, :escape => false }
54  end
55  -%>

57  <%= ## render :partial => 'sens_pełny_opis_link', :object =>
    [akat.id, token.id, @the_sens]
59  -%>
60  <%= render :partial => 'sens_long', :object => [akat.id, token.id,
    @the_sens]
61  -%>

63  <% end -%>

65  <% elsif akat_edytowalny?( :word_senses, akat ) -%>
66  <% # to się zapętli nieskończenie, jeśli dopuścimy oznaczenie jako
67      # zweryfikowanego akapitu, który nie ma określonych wszystkich
68      # sensów.
69      @the_sens = Sensy.new # dla radioguzika w poniższym partialu
70  -%>
71  <%= render :partial => 'token_wybs', :object => [akat, token,
    disamb_id, @the_sens.id] -%>
72  <% end # of if @the_sens
73  -%>
74      <% end # of do_sensu?
75      -%>

78  <%

```



```

79  ## # Local Variables:
80  ## # mode: mmm
81  ## # End:
82  -%>

```

## **\_token\_td.rhtml**

```

24  <% # wołany z
25      token, akat, lewy = token_td
26      akath = akat.akat_hash
27      eltid = "_#{akat.id}_#{token.id}"
28  -%>

30  <%= if akat_edytowalny?( :segmentation, akat, lewy ) and
31      token.sg_choice_id : "<div id=\"outer_seg-token#{eltid}\"" +
32      if token.segmentny?( akath ) :
33          "class=\"list-line-#{token.sg_variant_id % 2}\"" else "
34          class=\"list-line-rejected\""
35      end + ">"
36      elsif akat_edytowalny?( :sentences, akat, lewy )
37  <div id=\"outer_seg-token#{eltid}\">
38  end -%>

39  <% if lewy and @weryfikacja and token.id == @weryfikacja[:primafalsa]
40  ] -%>
41  <a name="primafalsa"></a>
42  <% end # of primafalsa
43  -%>

44  <% kończy_zdanie = (
45      if akat_edytowalny?( :sentences, akat, lewy)
46      KoniecZdaniaAnot.znajdz( akat, token.id )
47      else ( token.czy_konczy_zdanie? || KoniecZdaniaAnot.znajdz( akat,
48      token.id ) )
49  end
50  ) -%>

51  <% if false and akat_edytowalny?( :sentences, akat, lewy )
52  # 2009/6/28
53  # AP poprosił o usunięcie tego, jako zmniejszającego czytelność.
54  -%>

55  <%= link_to_remote (
56      if kończy_zdanie : 'nie tnij za mną'
57      else 'tnij za mną'
58      end ),
59      :url => {
60      :action =>:konczymy_zdanie,
61      :id => token.id,
62      :akat => akat,
63      :poziom =>:sentences, :escape => false } # ,
64      ## :update => 'disa-token' + token.id.to_s
65  -%>

```

```

68 <%= if r_zarzacca?
69     link_to_remote znak_zatwierdź_tę,
70         :url => {
71             :action => :sentences_zatwierdź_tę,
72             :id => token.id,
73             :akat => akat,
74             :poziom => :sentences, :escape => false }
75     end
76 -%>

79 <% end # of if :sentences
80 -%>

82 <%
83 if (not token.interp?( akath )) or
84 akat_edytowalny?( :segmentation, akat, lewy ) -%>
85 <i><%= h token.orth # + " (token.id)"
86 %></i><%=
87     if akat_edytowalny?( :segmentation, akat, lewy ) and
88         token.ns_następuje?
89         "<b>&#10216;bez spacji z nast.&#10217;</b>"
90     end %><% # (note the previous Ruby chunk was with =)
91     end # of if in l. 105
92 -%>

93 <% if not ( akat_edytowalny?( :segmentation, akat, lewy ) and
94     token.sg_choice_id )
95     alert = (( @weryfikacja and
96         @weryfikacja[ :morphosyntactic ] and
97         @weryfikacja[ :morphosyntactic ].include?( token.id ) ) or
98         ( token.superancja_include?( :morphosyntactic ) ))
99 -%>
100 <div id="disa-token<%=eltid %>"><%=
101     ## logger.info "##### token.id: #{token.id}, #{token.id}"
102     ## logger.info "##### @weryfikacja: " + @weryfikacja.inspect
103 ' class="alert-bg"' if alert %>>
104 <%
105     if akat_edytowalny?( :morphosyntactic, akat, lewy )
106         zmieniaj_dyzamb = (token.disamb( akath ).size != 1)
107         if zmieniaj_dyzamb
108             ( session[ :zmieniaj_dyzamb ] ||= Hash.new )[ token.id ] ||=
109                 true
110         end
111         zmieniaj_dyzamb = :tworz unless token.interpretacja[0]
112         zmieniaj_dyzamb = (zmieniaj_dyzamb or (
113             #
114             ( szd = session[:zmieniaj_dyzamb] ) and # the existence of
115             session[:zmieniaj_dyzamb] is not guaranteed: we create(d) it only if we
116             were to change the disamb.
117             szd[token.id] # it's nil if absent
118             so all right

```

```

116      ))
117      else zmieniaj_dyzamb = false
118      end # of if akat_edytowalny?
119      -%>

121      <% if zmieniaj_dyzamb
122          if DlaEli.nk and akat_edytowalny?( :morphosyntactic,
          akat, lewy )
123          if zmieniaj_dyzamb == :tworz
124              %>
125              <%= render :partial => 'token_tworz_tag', :object => [ token, akat]
              %>
126              <% else # not :tworz
127              -%>
128              <%= render :partial => 'token_wszi', :object => [akat, token] %>
129              <% end # of :tworz or not
130      else # not DlaEli.nk
131      -%>
132      <%= render :partial => 'token_wszi', :object => [akat, token] %>
133      <% end # of if nk or not
134      -%>
135      <% elsif # not zmieniaj_dyzamb
136      @poziomy.morphosyntactic? or alert or (
137      @poziomy.word_senses? and token.do_sensu?( akath ) )
138      # gdy jesteśmy na poziomie sensów, chcemy znać ms-tag
139      -%>
140      <%= render :partial => 'token_disamb', :object => [ token, akat,
          lewy ] %>
141      <% end # of if zmieniaj_dyzamb or not
142      -%>
143      </div> <% # of div disa-token
144      elsif token.interp?( akath ) -%>
145      <%= token.orth -%>
146      <% end # of conditions for showing morphosyntax.
147      -%>

149      <% if DlaEli.x or ( DlaEli.nk and @poziomy.word_senses? ) -%>
150      <div id="sens-anot<%=eltid %>"><%=
151          if (@weryfikacja and
152              @weryfikacja[:sensy] and
153              @weryfikacja[:sensy].include?(token.id) ) or
154              ( token.superancja_include?( :word_senses) )
155              ' class="alert-bg"'
156          end
157      %>>
158      <%= render :partial => 'token_sens', :object => [token, akat, lewy]
          %>
159      </div>
160      <% end # of if dla Eli orct.
161      -%>

```

```

163 <% alert = (( @weryfikacja and
164     @weryfikacja[:segmentation] and
165     @weryfikacja[:segmentation].include?( token.id ) ) or
166     ( token.superancja_include?(:segmentation)))
167     if ( akat_edytowalny?( :segmentation, akat, lewy ) or alert ) and
        token.sg_choice_id
168     sgvan = token.sg_variant_anot( akath )
169     ( session[ :zmieniaj_segmentacje ] ||= {} )[ token.id ] = true
        unless sgvan
170     -%>
171     <div id="segm-token<%= eltid %>"
172     <%= ' class="alert-bg"' if alert %>>
173     <%= render :partial => 'token_segm', :object => [akat, token, sgvan
174     ] -%>
175     </div>
176     <% end # of czy_segm?
        -%>

179 <% if czy_żle_granica_zdania =
180     (( @weryfikacja and
181         @weryfikacja[ :sentences ] and
182         @weryfikacja[ :sentences ].include?( token.id ) ) or
183         token.superancja_include?( :sentences ) )
184     -%>
185     <div id="sentence-token<%=eltid %>"<%= ' class="alert-bg"' %>>
186     <% end %>

188     <% if kończy_zdanie
189     -%>
190         (k.z.)
191     <% elsif czy_żle_granica_zdania -%>
192     &nbsp;
193     <% end # of czy kończy zdanie
194     -%>
195     <% if czy_żle_granica_zdania -%>
196     </div>
197     <% end %>

199 <%= if ( akat_edytowalny?( :segmentation, akat, lewy ) and
200     token.sg_choice_id ) or
201     akat_edytowalny?( :sentences,akat, lewy ): "</div>" end -%>

205 <%
206 ## # Local Variables:
207 ## # mode: mmm
208 ## # End:
209 -%>

```

### **\_token\_tworz\_tag.rhtml**

```

24 <% token, akat = token_tworz_tag
25     stt = session[:tworz_tag]

```

```

26     stt = stt[ token.id ] if stt
27     if stt
28         lemat, tag = stt[:lemat], stt[:tag]
29     else
30         lemat, tag = @lemat, nil
31     end
32     -%>

34     <% if flash[:tworz_tag] -%>
35     <div id="notice">
36         <%= flash[:tworz_tag] -%>
37     </div>
38     <% end -%>

41     <% remote_form_for :interpretacja_anot, @interpretacja_anot,
42         :url => {:action => :interpretacja_z_tagu,
43         :id => token.id , :poziom => :morphosyntactic,
44         :akat => @akapit_transzy,
45         :status => 0, :escape => false} do |form| -%>

47         <%= link_to_remote '-,-',
48             :url => {
49                 :action => :nowytag_skopiuuj_orth, :id => token.id,
50                 :akat => @akapit_transzy,
51                 :poziom => :morphosyntactic, :escape => false
52             }
53         -%>

55     <span id="<%= "token_lemat#{token.id}" %>">
56     <%= render :partial => 'tworz_tag_lemat', :object => [token.id, lemat]
57     -%>
58     </span>

59     <%= text_field "token_tag#{token.id}", 'tag', :value => tag, :size =>
60     25 %>

61     <div class="auto_complete"
62     id="<%= "\"token_tag#{token.id}_tag_auto_complete\""%>
63     </div>

65     <%= auto_complete_field "token_tag#{token.id}_tag",
66         :url=>{:action=>'autocomplete_tag', :id => token.id, :poziom=>
67         :morphosyntactic, :akat => @akapit_transzy, :escape => false },
68         :tokens => ', ',
69         :min_chars => -1 %>

71     <%= submit_tag "zatwierdź" , :class => "submit" -%>

73     <%= link_to_remote 'anuluj',
74         :url => {
75             :action => :dyszamb_anulowana, :id => token.id,
76             :akat => @akapit_transzy,
77             :poziom => :morphosyntactic, :escape => false
78         }

```

```

79      -%>

81    <% end # of form
82      -%>

_token_wszih.html

24    <!-- wywołujemy go dla obiektu token klasy Token -->

27    <% akat, tok = token_wszih
28      @interpretacja_anot = tok.disamb(session[:uzid])[0]
29      -%>

32      <fieldset>
33        <% remote_form_for :interpretacja_anot, @interpretacja_anot,
34          :url => { :action => :nowa_dyzambiguacja,
35            :id => tok.id ,
36            :status => 0,
37            :akat => akat,
38            :poziom => :morphosyntactic, :escape => false } do |form| -%>

41    <% tok.interpretacja.each {|interpr| -%>
42    <%= radio_button :interpretacja_anot, :interpretacja_id,
43      interpr.interpretacja_id,
44      { :onclick =>
45        ajax_request("/#{annotation_controller}/nowa_dyzambiguacja/#{tok.id}?status=0&p
46      %>

48    <%= "<b>" if interpr.disamb? -%>
49    <i><%= interpr.leksem.lemat -%></i>
50    <%= interpr.fulltag -%>
51    <% if tok.interpretacja.index(interpr)+1<tok.interpretacja.size
52      -%>
53    <br>
54    <% end -%>
55    <%= "</b>" if interpr.disamb? -%>
56    <% } -%>

58    <%= ## submit_tag "zatwierdź" , :class => "submit"
60      -%>

62    <% if DlaElLink -%>
63    <%= link_to_remote 'dodaj',
64      :url => {
65        :action => :token_tworz_tag,
66        :id => tok.id,
67        :akat => akat,
68        :poziom => :morphosyntactic, :escape => false },
69    :update => "disa-token_#{akat.id}_#{tok.id}" -%>
70    <% end -%>

72    <%= link_to_remote 'anuluj',
73      :url => {
74        :action => :dyzamb_anulowana, :id => tok.id, :akat => akat,
        :escape => false

```

```

75     # :status => params[:status]
76   }
77   -%>

```

```

79   <% end # of form
80   -%>
81   </fieldset>

```

## **\_token\_wybs.rhtml**

```

24   <% # wywołujemy go z tablicą [akat, token]
25   # z partiala token_sens wołanego z widoku anotuj.
26   akat, tok, disaid, the_sens_id = token_wybs -%>

28   <fieldset>
29     <% remote_form_for :sens_anot, the_sens_id,
30     :url => { :action => :wybor_sensu,
31       :id => tok.id,
32       :akat => akat,
33       :poziom => :word_senses, :escape => false
34     } do |form| %>

36   <% ts187 = tok.sensy( akat.akat_hash )
37   ts187.each {|sens| -%>
38     <%= radio_button :the_sens, # nazwa zmiennej instancyjnej
39       (@the_sens)
40       :sens_id, # metoda tej zmiennej instancyjnej (akcesor atrybutu)
41       sens.id, # wartość
42       { :onclick =>
43         ajax_request( "#{annota-
44           tion_controller}/wybor_sensu/#{tok.token_id}?poziom=word_senses&akat=#{akat.id}&disamb_id=#
45           }
46         %>
47       <%= # "<b>" if interpr.disamb?
48       -%>
49       <%= # sens.id AP nie chce.
50       -%>
51       <%= sens.short_description -%>
52       <%= render :partial => 'sens_pełny_opis_link', :object => [akat.id,
53         ts187.id, sens]
54       -%>
55       <%= render :partial => 'sens_long', :object => [akat.id, ts187.id,
56         sens]
57       -%>
58       <% if true or ts187.index( sens )+1< ts187.size
59       -%>
60       <br>
61       <% end -%>
62       <%= # "</b>" if interpr.disamb?
63       -%>
64     } -%>

```

```

63      <%= ## submit_tag "&#10004;" , :class => "submit"
64      # niepotrzebne i mylące.
65      -%>

68      <%= link_to_remote '&#10007;',
69      :url =>{
70          :action =>:sensowanie_anulowane, :id => tok.id, :akat => akat,
71          :escape => false } ## ,
72      ##      :update => "sens-anot_#{akat.id}_#{tok.id}"
73      -%>

77      <% end # of form
78      -%>
79      </fieldset>

```

### **\_tworz\_tag\_lemat.rhtml**

```

24      <% tokid, lemat = tworz_tag_lemat -%>

26      <%= text_field "token_lemat#{tokid}", 'lemat', :value => lemat,
27      :size => 17 %>

```

### **\_wsze\_transze\_BW.rhtml**

```

24      <div id="wsze_transze_otwarte">

26      <h1> <%= Transza::HOW_MANY_RECENT %> ostatnio zakończonych transz (dla
      Beaty) </h1>

28      Transze są pogrupowane wg loginów, a w tych grupach – od najświeższej.

31      <% prev_login = nil
32      Transza.recent_closed.each { |tra| %>
33      <%
34      if prev_login != tra.login %>

36      <%= "</ul>" if prev_login %>
37      <% prev_login = tra.login %>
38      <h4> <%= tra.login %> </h4>
39      <ul>
40      <% end %>
41      <li><b><%= tra.id %></b> <%= tra.true_updated_at %>

43      <% } %>

46      <h1> (dla Beaty, lepiej nie używać) Wszystkie transze, najpierw
47      otwarte, potem zamknięte, wg Anotatorów </h1>

49      <% if session[ :pokaż_wsze_transze_BW ] -%>

51      <p><%= link_to_remote "ukryj" , :url => { :action =>
      'wsze_transze_BW_toggle',
52      :direction => 'hide' } %>
53      </p>

55      <p> Razem „w grze” (czyli co najmniej pobranych przez anotatorów):
56      <%= AkapitTranszy.ile_zaanotowanych( nil, :zakończony ) %>

```



```

57     zakończonych,
58     <%= AkapitTranszy.ile_zaanotowanych( nil, :oczekujące ) %>
59     oczekujących na drugiego –
60     <%= AkapitTranszy.ile_zaanotowanych( nil, :anotaśne ) %> pobranych
61     nadających się do anotacji.
62     </p>

64     <p>
65     »zak./pods.« jest skrótem na ‘zakończonych lub podsądnych’ – chodzi o
66     liczbę transz, w&nbsp;których Anotator/ka zrobił/a już wszystko, co
67     w&nbsp;jej/go mocy.
68     </p>

71     <% Transza.wsze.each do |anor_transze|
72         ## wsze_otwarte
74         # to jest tablica [ [anotator, jego-transze-otwarte]<*> ]
75         -%>
76         <h3><%= anor_transze[0].login %>:</h3>
77         <b><%= anor_transze[0].ile_transz_zakończonych( :morphosyntactic )
78         %></b>
79         tr. zak./pods. na poz. morfoskładniowym; &nbsp; &nbsp;
80         <b><%= anor_transze[0].ile_transz_zakończonych( :word_senses ) %></b>
81         tr. zak./pods. na poz. sensów słów. <br/>

82         <%= anor_transze[0].ile_zakończonych %> akap. zakończone/ych,
83         <%= anor_transze[0].ile_oczekujących %> oczekujące/ych na drugiego
84         – <%= anor_transze[0].ile_anotaśnych %>
85         do anotacji.
86         <%= render :partial => 'lista_transz', :object => anor_transze[1] %>
87         <% end %>

89         <p><%= link_to_remote "ukryj" , :url => { :action =>
90         'wsze_transze_otwarte_toggle',
91         :direction => 'hide' } %>
92         </p>

93         <% else # nie pokazuj wszystkich otwartych
94         -%>
95         <p><%= link_to_remote "pokaż" , :url => { :action =>
96         'wsze_transze_BW_toggle',
97         :direction => 'show' } %>
98         Uwaga! Renderuje się kilkadziesiąt sekund!
99         </p>

101         <% end # of pokazuj wszystkie otwarte lub nie pokazuj
102         -%>

104     </div>

106     <%
107     ## # Local Variables:
108     ## # mode: mmm
109     ## # End:
110     -%>

```

**\_wsze\_transze\_otwarte.rhtml**

```

24   <div id="wsze_transze_otwarte">
26   <h1> Transze przydzielone i nie zakończone pogrupowane wg Anotatorów
    </h1>
28   <% if session[ :pokaż_wsze_transze_otwarte ] -%>
30   <p><%= link_to_remote "ukryj" , :url => { :action =>
    'wsze_transze_otwarte_toggle',
31   :direction => 'hide' } %>
32   </p>
34   <p> Razem „w grze” (czyli co najmniej pobranych przez anotatorów):
35   <%= AkapitTranszy.ile_zaanotowanych( nil, :zakończony ) %>
36   zakończonych,
37   <%= AkapitTranszy.ile_zaanotowanych( nil, :oczekujące ) %>
38   oczekujących na drugiego –
39   <%= AkapitTranszy.ile_zaanotowanych( nil, :anotaśne ) %> pobranych
40   nadających się do anotacji.
41   </p>
43   <p>
44   »zak./pods.« jest skrótem na ‘zakończonych lub podsadnych’ – chodzi o
45   liczbę transz, w&nbsp;których Anotator/ka zrobił/a już wszystko, co
46   w&nbsp;jej/go mocy.
47   </p>
50   <% Transza.wsze_otwarte.each do |anor_transze|
51     ## wsze_otwarte
53     # to jest tablica [ [anotator, jego-transze-otwarte]<(*)> ]
54     -%>
55     <h3><%= anor_transze[0].login %></h3>
56     <b><%= anor_transze[0].ile_transz_zakończonych( :morphosyntactic )
    %></b>
57     tr. zak./pods. na poz. morfoskładniowym; &nbsp;&nbsp;&nbsp;
58     <b><%= anor_transze[0].ile_transz_zakończonych( :word_senses ) %></b>
59     tr. zak./pods. na poz. sensów słów. <br/>
61     <%= anor_transze[0].ile_zakończonych %> akap. zakończone/ych,
62     <%= anor_transze[0].ile_oczekujących %> oczekujące/ych na drugiego
63     – <%= anor_transze[0].ile_anotaśnych %>
64     do anotacji.
65     <%= render :partial => 'lista_transz', :object => anor_transze[1] %>
66     <% end %>
68   <p><%= link_to_remote "ukryj" , :url => { :action =>
    'wsze_transze_otwarte_toggle',
69   :direction => 'hide' } %>
70   </p>
72   <% else # nie pokazuj wszystkich otwartych

```

```

73     -%>
74     <p><%= link_to_remote "pokaż" , :url => { :action =>
75         'wsze_transze_otwarte_toggle',
76         :direction => 'show' } %>
77     Uwaga! Renderuje się kilkadziesiąt sekund!
78     </p>
79
80     <% end # of pokazuj wszystkie otwarte lub nie pokazuj
81     -%>
82
83     </div>
84
85     <%
86     ## # Local Variables:
87     ## # mode: mmm
88     ## # End:
89     -%>

```

### \_wsze\_transze\_otwarte\_BW.rhtml

```

24     <div id="wsze_transze_otwarte">
25
26     <h1> 40 ostatnio zakończonych transz (dla Beaty) </h1>
27
31     <h1> Wszystkie transze, najpierw otwarte, potem zamknięte, wg
32     Anotatorów (dla Beaty) </h1>
33
34     <% if session[ :pokaż_wsze_transze_BW ] -%>
35
36     <p><%= link_to_remote "ukryj" , :url => { :action =>
37         'wsze_transze_BW_toggle',
38         :direction => 'hide' } %>
39     </p>
40
41     <p> Razem „w grze” (czyli co najmniej pobranych przez anotorów):
42     <%= AkapitTranszy.ile_zaanotowanych( nil, :zakończony ) %>
43     zakończonych,
44     <%= AkapitTranszy.ile_zaanotowanych( nil, :oczekujące ) %>
45     oczekujących na drugiego –
46     <%= AkapitTranszy.ile_zaanotowanych( nil, :anotaśne ) %> pobranych
47     nadających się do anotacji.
48     </p>
49
50     <p>
51     »zak./pods.« jest skrótem na ‘zakończonych lub podsądnych’ – chodzi o
52     liczbę transz, w&nbsp;których Anotator/ka zrobił/a już wszystko, co
53     w&nbsp;jej/go mocy.
54     </p>
55
56     <% Transza.wsze.each do |anor_transze|
57         ## wsze_otwarte
58         # to jest tablica [ [anotator, jego-transze-otwarte]<*> ]
59         -%>
60
61     <h3><%= anor_transze[0].login %>:</h3>
62     <b><%= anor_transze[0].ile_transz_zakończonych( :morphosyntactic )
63     %></b>

```

```

63 tr. zak./pods. na poz. morfoskładniowym; &nbsp; &nbsp; &nbsp;
64 <b><%= anor_transze[0].ile_transz_zakończonych( :word_senses ) %></b>
65 tr. zak./pods. na poz. sensów słów. <br/>

67 <%= anor_transze[0].ile_zakończonych %> akap. zakończone/ych,
68 <%= anor_transze[0].ile_oczekujących %> oczekujące/ych na drugiego
69 - <%= anor_transze[0].ile_annotaśnych %>
70 do anotacji.
71 <%= render :partial => 'lista_transz', :object => anor_transze[1] %>
72 <% end %>

74 <p><%= link_to_remote "ukryj" , :url => { :action =>
'wsze_transze_otwarte_toggle',
75 :direction => 'hide' } %>
76 </p>

78 <% else # nie pokazuj wszystkich otwartych
79 -%>
80 <p><%= link_to_remote "pokaż" , :url => { :action =>
'wsze_transze_otwarte_toggle',
81 :direction => 'show' } %>
82 Uwaga! Renderuje się kilkadziesiąt sekund!
83 </p>

86 <% end # of pokazuj wszystkie otwarte lub nie pokazuj
87 -%>

89 </div>

91 <%
92 ## # Local Variables:
93 ## # mode: mmm
94 ## # End:
95 -%>

```

### anotuj.rhtml

```

24 <div id="fixed_hdr">
25   <table class="table-plain">
26     <%
27       ##          logger.info "@@@ anotuj.rhtml w. 4. #{Time.now}"
29       if DlaEli.nk -%>
30         <tr>
31           <td colspan="2">
32             <% form_for :poziomy_anotacji, @poziomy_wybrane, :url =>
33               {:action => :poziomuj, :akat => @akapit_transzy,
34               :poziom => :any, :escape => false } do |form| %>
35               <%= submit_tag "Poziomy anotacji" %>
36               &nbsp; &nbsp; &nbsp;
38             <% PoziomyAnotacji.działające.each do |poziom|
39               ## unless poziom == :segmentation and
41               ## @akapit_transzy.ma_status?( :segmentation, :>=, :zweryfikowany
               )

```

```

43   -%>
44   <%= form.check_box poziom, {}, true, false
45   %>&nbspsp;=<%= PoziomyAnotacji.nom( poziom ) -%>
46   &nbspsp;
47   <% ## end # of unless segmentation.
48   end # of each poziom.
49   -%>
50
51   <% end # of form.
52   -%>
53
54   <% if r_zarzadca? -%>
55   <% form_for :jak_anotator, @uzytkownik, :url =>
56   {:action => :zarzadca_jak_anotator,
57   :akat => @akapit_transzy,
58   :poziom => :any, :escape => false } do |form| %>
59   <%= submit_tag "Zatwierdź" %>
60   &nbspsp;
61
62   <%= form.check_box :jak_anotator,
63   { }, true, false -%>
64   jak anotator
65   <% end # of form
66   end # of if zarzadca
67   -%>
68       </td>
69     </tr>
70     <% end # of if nk.
71     -%>
72     <tr>
73       <td colspan="2">
74         <div class="between-dots">
75           <div class="akapit-overflow">
76             <span style="font-size: xx-small"><%= @akapit.akapit_id %>
77             (tr.&#8201;<%= @akapit_transzy.transza_id %>, at.&#8201;<%=
              @akapit_transzy.id %>)).</span>
78             <%= @akapit.tresc %>
79           </div>
80         </div>
81       </td>
82     </tr>
83     <tr>
84       <td id="status_info">
85         <%= render :partial => 'status_info', :object => @akapit_transzy
86         -%>
87       </td>
88       <td id="zatwod_buttons" align="right">
89         <%= render :partial => "anotuj_zatwod_nk" -%>
90       </td>
91     </tr>
92   </table>

```

```

93   </div>

95   <div id="main1" >

97     <%= render :partial => '/layouts/notice' -%>

100    <!-- przerobione /app/views/akapit/show.rhtml -->

102    <table>
103      <% if @werdykt == -1or r_zarzacda? -%>
104        <tr>
105          <td colspan="<%=
106            @poziomy.colspan + ( if @do_poprawki : 0 else 2 end )
107            %>">
108            <%= uz_login( @anotid ) %> </td>
109            <% unless @do_poprawki -%>
110              <td colspan="<%=
111                @poziomy.colspan
112                %>">
113              <%= render :partial => 'blizniaczy_status' %>
114              </td>
115            <% end # of unless @do_poprawki
116            -%>
117          </tr>
118          <% end # of if @werdykt ==-1 or r_zarzacda?
119          %>

122    <%
123    ##      logger.info "@@@ anotuj.rhtml w. 105. #{Time.now}"
125    if akat_edytowalny?( :sentences, @akapit_transzy )
126      -%>
127      <p id="sentensnik" style="line-height:230%">
128      <%= render :partial => 'sentensnik', :object => @akapit_transzy -%>
129      </p>
130      <% end # of if edytowalny :sentences
131      -%>

134      <% @akapit.token.each do |token|>
135      <tr valign="top" class="<%= cycle('list-line-odd',
136        'list-line-even') %>" >
137        <td>&#8226;</td>
138        <td>
139        <%= render :partial => 'token_td', :object => [ token,
140          @akapit_transzy, true ] -%>
141        </td>
142        <% if (not DlaEli.nk) or @poziomy.syntactic? -%>
143        <%= render :partial => 'token_framka', :object => [token,
144          @akat_hash] -%>
145        <%= render :partial => 'token_fra', :object => [token,
146          @akat_hash, :lewa] -%>
145        <% end# of if syntactic?
146        -%>

```

```

149      <% if (@werdykt == -1 and @pokaż_bliż) or r_zarzadca? -%>
150      <td class="gruba-kreska"></td>

152      <td>
153      <%= render :partial => 'token_td', :object => [token,@akat_bliźniak,
false] -%>
154      </td>

156      </td>

158      <% if (not DlaEli.nk) or @poziomy.syntactic? -%>
159      <%= render :partial => 'token_framka', :object => [token,
@bliakath] -%>
160      <%= render :partial => 'token_fra', :object => [token, @bliakath]
-%>
161      <% end # of if nk.syntactic?
162      -%>

164      <% end # of if @werdykt== -1 and @pokaż_bliż .
165      -%>

167      </tr>
168      <% end # of token.each
169      ## logger.info "@@@ anotuj.rhtml w. 157. #{Time.now}"
171      -%>
172      <tr>
173      <td></td>
174      <td>
175      </td>
176      <% if @werdykt == -1 or r_zarzadca?-%>
177      <%= if DlaEli.nk
178      "<td colspan=#{@poziomy.colspan+1}\>"></td>"
179      else "<td></td>"
180      end %>
181      <% end -%>
182      <% if (not DlaEli.nk) or @poziomy.syntactic? -%>
183      <td></td><!-- tylko dla Eli?-->
184      <td>
185      <% if akat_edytowalny?( :syntactic ) -%>
186      <%= button_to('wyczyść frazy', { :action => 'frazy_wyczysc',
187      :poziom => :syntactic, :akat => @akapit_transzy},
188      {:confirm => "Czy rzeczywiście chcesz wyczyścić wszystkie frazy?",
189      :method => :post}) -%>
190      <% end -%>
191      </td>
192      <% end # of if syntactic
193      -%>
194      </tr>
195      </table>

197      <% if Walencja.x -%>
198      <!-- walencje-->
199      <div id="walencja">

```

```

200     <%= render :partial => 'walencja' -%>
201   </div>
202   <% end -%>

205   <!-- komentarze-->

207   <h3> Komentarze, prośby </h3>
208   <%= render :partial => 'komentarze',
209     :object => [ @akapit_transzy, :anotacja ] -%>

211   <%= render :partial => 'komentarz_dodaj', :object =>
    [@akapit_transzy, nil] -%>

213   <div id="prosby_annotatorek">
214     <%= render :partial => 'prosby_annotatorek' -%>
215   </div>

217   <%
218   ##       logger.info "@@@ anotuj.rhtml w. 204. #{Time.now}"
220   %>

222   <%
223   ## # Local Variables:
224   ## # mode: mmm
225   ## # End:
226   -%>

```

### autocomplete\_tag.rhtml

```

24   <ul class="autocomplete_list" >
25     <% @tags.each do |t| %>
26       <li class="autocomplete_item" ><%= t %></li>
27     <% end %>
28   </ul>

```

### lista\_transz.rhtml

```

24   <div id="main">
25     <%= render :partial => '/layouts/notice' -%>
26     <%= render :partial => '/layouts/powitalny' -%>

28     <% unless DlaEli.nk -%>
29     <p>
30     Na konfrontację oczekuje/a
31     <%= ZdaniaTranszy.ile_zaanotowanych(session[:uzid], :do_konfrontacji)
32     %> zdanie/nia/ń.
33     </p>
34     <% end # of unless nk
35     -%>

38     <%= render :partial => 'layouts/bug_reports' -%>

40     <h1>Lista transz</h1>

43     <%= h(@anotator.login) %>, masz przypisane i nie zakończone:

```



```

45   <div id="lista_transz">
46   <%= render :partial => 'lista_transz', :object =>
      @anotator.transze_otwarte %>
47   </div>

50   <% if DlaEli.x -%>
51   <div id="odbierz_transze">
52     <%= render :partial => 'prosba_link', :object =>
      @anotator.uzytownik_id -%>
53   </div>

55   <div id="lista_odebranych">
56   <%= render :partial => 'lista_odebranych', :object =>
      @odebrania_transz -%>
57   </div>
58   <% end # of if dla Eli
59   -%>

62   <% if session[:transze_zamkniete_pokaz] %>
63   <h1> Transze zakończone </h1>
64   <%= link_to '[ukryj transze zakończone]',
65   {:action => 'transze_zamkniete_ukryj'}, {} %>
66   <%= render :partial => 'lista_transz', :object =>
      @anotator.transze_zamkniete %>

68   <% elsif @sa_transze_zamkniete # not session[:transze_zamkniete_pokaz]
69   %>
70   <div>
71   <%= link_to '[pokaż transze zakończone]',
72   {:action => 'transze_zamkniete_pokaz'}, {} %>
73   </div>
74   <% end # of pokaz/ukryj tr. zamknięte
75   %>
76   </div>

78   <%
79   ## # Local Variables:
80   ## # mode: mmm
81   ## # End:
82   -%>

```

### nie\_chce\_transzy.rhtml

```

24   <div id="main">
25   <%= render :partial => '/layouts/notice' -%>

27   <div id="odbierz_transze">
28     <%= render :partial => 'prosba_link', :object =>
      @anotator.uzytownik_id -%>
29   </div>

31   <div id="lista_odebranych">
32   <%= render :partial => 'lista_odebranych', :object =>
      @odebrania_transz -%>
33   </div>

```

**ogladactwo.rhtml**

```

24  <% log_time "ogladactwo 1" -%>

26  <div id="main">
27    <div id="flash_notice">
28      <%= render :partial => '/layouts/notice' -%>
29    </div>
30    <%= render :partial => '/layouts/powitalny' -%>
31    <p>

33    <% if Rola.wszewid?(session[:rola_id]) -%>
34    Rozbieżności występują w&nbsp;<%= @ile_rozbieznych %>
35      akapicie/tach. <br/>
36    Do osądzenia dojrzało/y <%= @ile_podsadnych %>.
37    </p>

39    <% if @ile_podsadnych_total != 0 -%>

41    <h1>Przejdź do </h1>
42    <%= render :partial => 'ogladactwo_goto' -%>

45    <h1>Akapity podsądne</h1>

47    <%= render :partial => 'ogladactwo_lista_akapitow', :object =>
48    [@akapity_podsadne, if Rola.audytork?(session[:rola_id]) : "Doradzaj"
49    else "Rozsądzaj" end] -%>

51    <% end # of if są rozbieżne.
52    -%>

54    <% log_time "ogladactwo po podsadnych przed skoment. (27)" -%>

56    <div id="akapity_skomentowane">
57      <%= render :partial => 'akapity_skomentowane' -%>
58    </div>

60    <% log_time "ogladactwo 32" -%>

62    <div id="prosby_annotatorek">
63      <%= render :partial => 'prosby_annotatorek' -%>
64    </div>

66    <div id="wsze_transze_BW">
67      <%= render :partial => 'wsze_transze_BW'
68      -%>
69    </div>

71    <div id="wsze_transze_otwarte">
72      <%= render :partial => 'wsze_transze_otwarte'
73      -%>
74    </div>

77    <% else # of if wszewid .
78    -%>

```

```

80  <h1>Akapity zweryfikowane</h1>
82  <%   qtra = 100
83      (Akapit.min_id/qtra).upto( Akapit.max_id/qtra + 1)  do |id|
84          limits = [id* qtra+1, (id+1)* qtra]
85          quasi_transza = Akapit.quasi_transza( limits )
86      if quasi_transza.size > 0 -%>
87          <div id="quasi_transza_<%= limits.join("-") %>">
88              <%= render :partial => 'quasi_transza', :object => [quasi_transza,
89                  limits.join("-")] -%>
90          </div>
91          <% end # of quasi_transza.size,
92          end # of min_id.upto( max_id ).
93          -%>
94      <% end # of if wszewid?.
95      -%>
96
97  </div>
98
101  <% log_time "ogładactwo 65" -%>
102  <%
103  ## # Local Variables:
104  ## # mode: mmm
105  ## # End:
106  -%>

```

### **pokaz\_transze.rhtml**

```

24  <div id="fixed_hdr">
25  <h1>Lista akapitów transzy <%= @transza.opis %></h1>
26  </div>
27
28  <div id="main1">
29
30  <%= render :partial => '/layouts/notice' -%>
31
32  <%= render :partial => 'lista_akapitow', :object => @akapity -%>
33
34  <br />
35  </div>
36
37  <%
38  ## # Local Variables:
39  ## # mode: mmm
40  ## # End:
41  -%>

```

### **quasi\_transza\_brutal.rhtml**

```

25  <div id="main">
26  <div id="flash_notice">
27  <%= render :partial => '/layouts/notice' -%>
28  </div>
29
30  <h1>Akapity zweryfikowane <%= @klucz %></h1>

```

```

32 <%= link_to "[lista akapitów]",
33       :action => :zweryf
34 -%>

36 <%= render :partial => 'ogladactwo_lista_akapitow', :object
   =>[Akapit.quasi_transza( @qt_klucz ), :Oglądaj, :no_inf] -%>

38 <%= link_to "[lista akapitów]",
39       :action => :zweryf
40 -%>

43 </div>

```

### sense\_inventory.rhtml

```

24 <div id="main1" >

26   <%= render :partial => '/layouts/notice' -%>

29   <% ukryj_wsio = link_to "ukryj pełne opisy" , :action =>
      'sense_long_hide_all' -%>
30   <%= ukryj_wsio %>

32   <% CzMLEksem.find( :all, :order => "cz_m_leksem.xmlid",
33       :include => :sensy ).each { |czml| %>

35     <h3><%= czml.lemat %></h3>
36     <ul>
37       <% czml.sensy.each { |sens| %>
38         <li>
39           <%= sens.short_description -%>
40           <%= render :partial => 'sens_pełny_opis_link', :object => [666,
41             66660000 + sens.id, sens]
42           -%>
43           <%= render :partial => 'sens_long', :object => [666, 66660000
44             + sens.id, sens]
45           -%>
46         } # of each sens
47       %>

49     </ul>
50     <% } # of each czmleksem
51     %>

53   <%= ukryj_wsio %>

55 </div>

```

### wyszukaj\_akapity.rhtml

```

24 <div id="fixed_hdr">
25 <h1>Wyszukiwanie akapitów wg lematu, sensu i klasy gramatycznej</h1>
26 </div>

28 <div id="main1">

```

```

30  <%= render :partial => '/layouts/notice' -%>

32  <fieldset>
33    <legend>Zapytanie</legend>
34    <% form_for :szukaj, :url => {:action => :wyszukaj_akapity } do
      |form| %>
35      <p>
36        <label for="szukaj_lemat" >Lemat(y):</label>
37        <%= form.text_field :szukaj_lemat, :value =>
38          session[:szukaj_lemat], :size => 40 -%>
39        </p>
40        <p>
41          <label for="szukaj_sens" >Sens(y):</label>
42          <%= form.text_field :szukaj_sens, :value => session[:szukaj_sens],
            :size =>40 -%>
43          </p>
44          <p>
45            <label for="szukaj_klasa" >Klasa(y) gramatyczna(e):</label>
46            <%= form.text_field :szukaj_klasa, :value =>
              session[:szukaj_klasa], :size =>40 -%>
47            </p>
48            <%= submit_tag "zapytaj" , :class => "submit" %>
49            <%= submit_tag "zawęż poprzednie" , :class => "submit" %>

51      <% end # of form.
52      -%>
53  </fieldset>

57  <% if @akapity -%>
58  <br/>
59  <%= @akapity.size %> akapit/y/ów transzy.
60  <br/><br/>
61  <%= render :partial => 'lista_akapitow', :object => @akapity -%>
62  <% end # of if zdania.
63  -%>
64  <br />
65  </div>

68  <%
69  ## # Local Variables:
70  ## # mode: mmm
71  ## # End:
72  -%>

```

## zweryf.rhtml

```

24  <div id="main">
25    <div id="flash_notice">
26      <%= render :partial => '/layouts/notice' -%>
27    </div>

29  <h1>Akapity zweryfikowane</h1>

```

```

32 <%= link_to "[schowaj wszystkie]",
33       :action => :ukryj_wszystkie_quasi_transze
34 -%>

37 <%   qtra = 100 # Transza::LICZBA_AKAPITÓW 2010/3/12 zgodnie
    z sugestią Superów zmieniona na 100.
38   (Akapit.min_id/qtra).upto( Akapit.max_id/qtra + 1) do |id|
39     limits = [id* qtra+1, (id+1)* qtra]
40     quasi_transza = Akapit.quasi_transza( limits )
41   if quasi_transza.size > 0 -%>
42     <div id="quasi_transza_<%= limits.join("-") %>">
43     <%= render :partial => 'quasi_transza', :object => [quasi_transza,
44       limits.join("-")] -%>
45   </div>
46   <% end # of quasi_transza.size
47   end # of min_id upto max_id
    -%>

50 <%= link_to "[schowaj wszystkie]",
51       :action => :ukryj_wszystkie_quasi_transze
52 -%>

55 </div>

```

## app/views/aux

### sensy.rhtml

```

24 <div id="main">
25 <h1>Tabele sensów</h1>
26 <p> Ctrl+F aby wyszukać właściwy leksem. </p>
27 <p> Kolumna anot_ct zawiera liczbę użyć danego sensu przez Anotatorów
28   (liczoną wg zdań transzy). Kolumna zweryf_ct zawiera liczbę wystąpień
29   danego sensu w zdaniach zakończonych (liczoną wg liczby zdań). </p>

31 <h3> Sensy rzeczowników </h3>

33 <table>
34   <% @nounsenses.each do |ns| -%>
35     <tr>
36       <% ns.each do |nscell| -%>
37         <td><%= nscell %> </td>
38       <% end end -%>
39     </tr>
40   </table>

43 <h3> Sensy czasowników </h3>

45 <table>
46   <% @verbsenses.each do |ns| -%>
47     <tr>
48       <% ns.each do |nscell| -%>
49         <td><%= nscell %> </td>

```

```

50      <% end end -%>
51      </tr>
52    </table>
53
54  </div>
55

```

**app/views/debug**

**lista sciezek.rhtml**

```

24 <ul>
25 <%
26     cuś_jest = false
27     Path.find_by_sql( " select * from path where " +
28         " path_id in ( select path_id from akapit " +
29         " if @to_są_akapity : " where akapit_id "
30         else " inner join akapit_transzy using( akapit_id)  where
31         akapit_transzy_id "
32     end +
33     " if @between211 : @lista211
34     else " in ({@lista211})"
35     end +
36     ") order by path_id "
37     ).each { |p211|
38     cuś_jest = true unless cuś_jest
39     -%>
40     <li> ścieżka <%= p211.id %>: &nbsp;&nbsp;&nbsp;<%= p211.path_text %>
41     <% } %>
42     <% unless cuś_jest %>
43     <li> takjigo tu ni ma, Panocku
44     <% end %>
45 </ul>
46
47 <%
48 ## # Local Variables:
49 ## # mode: mmm
50 ## # End:
51 -%>

```

**lista transz.rhtml**

```

24  <p>
25  <%
26      cuś_jest = false
27      Path.find_by_sql( " select distinct transza_id from akapit_transzy
        where " +
28      if @to_są_akapity : " akapit_id "
29      else " akapit_transzy_id "
30      end +
31      if @between211 : @lista211

```

```

32     else " in ({@lista211})"
33     end +
34     " order by transza_id "
35     ).each { |ti211|
36     cuś_jest = true unless cuś_jest
37     -%>
38     <%= ti211.transza_id %>,
39     <% } %>
41     <% unless cuś_jest %>
42     <li> takjigo tu ni ma, Panocku
43     <% end %>
48 <%
49 ## # Local Variables:
50 ## # mode: mmm
51 ## # End:
52 -%>

```

### **\_mwa\_ha\_ha.rhtml**

```

24 <h3> Mwa ha ha, I am Satan, Mwa ha ha! <br/>
25 <%= mwa_ha_ha %>
26 </h3>

```

### **debug-trash.rhtml**

```

25 <%= text_field_with_auto_complete :interpretacja, :reszta_tagu, {},
26 :skip_style => true %>
29 <%
30 token_id = 39665
31 token = Token.find( token_id )
32 lemat = token.interpretacja[0].leksem.lemat
33 stt = session[:tworz_tag]
34 if stt
35   lemat, tag = stt[:lemat], stt[:tag]
36 else
37   tag = nil
38 end
40 remote_form_for :interpretacja_anot, @interpretacja_anot,
41 :url => { :controller => :anotacja, :action =>
42 :interpretacja_z_tagu,
43 :id => token_id ,
44 :status => 0} do |form| -%>
45 <%= text_field "token#{token_id}", 'lemat', :value => lemat,
46 :size => 20 %>
49 <%= text_field "token#{token_id}", 'tag', :value => tag, :size => 25 %>
51 <div class="auto_complete"
52 id=<%= "\ " token#{token_id}_tag_auto_complete\ " %>>

```



```

53   </div>

55   <%= auto_complete_field "token#{token_id}_tag",
56     :url=>{:controller =>:anotacja, :action=>'autocomplete_tag', :id
      => token_id},
57   :tokens => ', ',
58   :min_chars => -1 %>

60   <%= submit_tag "zatwierdź" , :class => "submit" -%>

62   <%= link_to_remote 'anuluj',
63     :url => {:controller =>:anotacja,
64       :action =>:dyzamb_anulowana, :id => token_id
65   }
66   -%>

68   <% end # of form
69   -%>

72   <% form_for :interpretacja, :url => { :action => :create } do |form| %>
73   <p>Title: <%= form.text_field :title, :size => 30 %></p>
74   <p>Description: <%= form.text_area :description, :rows => 3 %></p>
75   <p>Image URL: <%= form.text_field :image_url %></p>
76   <p>Price: <%= form.text_field :price, :size => 10 %></p>
77   <%= form.select :title, %w{ one two three } %>
78   <p><%= submit_tag %></p>
79   <% end %>

84   <h3> Sensy akapitu 3576 </h3>

86   <% a = Akapit.find( 3576 )
87   akat = a.akapit_transzy[0] -%>
88   <p><%= a.tresc -%></p>

90   <% akath = akat.akat_hash
91     z.token.each do |tok|
92       s1 = tok.sens( akath )
93       if s1
94       -%>
95       <%= s1.leksem.lemat + ": " + s1.sens_anot_count.to_s -%> <br/>

97   <% end # of if s
98   end # of do |tok|
99   -%>

101  <p> Akapit 3258 </p>

103  <% Akapit.find(3258).akapit_transzy.each do |akat| %>
104  <%= akat.transza_auto_id %>
105  <% ui=akat.get_uzid %>
106  <%= ui %> <%= Uzytkownik.find(ui).login %> <br>
107  <% end %>

110  <p> akapit transzy i uzytkownicy interpretacji 175206 </p>
112  <% z=Interpretacja.find(175206).token.akapit %>

```

```

113 <%= z.akapit_id %> <%= z.tresc %>
114 <% z.akapit_transzy.each{ |akat| %>
115     <%= akat.get_uzid %> <%= akat.transza_auto_id %> <br>
116 <% } %>

119 <h3> Ile sensów anotowanych dla „bronić”</h3>

121     <%= Leksem.find(:all, :conditions =>{:lemat => 'bronić'})>
122     |lex|
123     lex.sensy_leksemu.collect{ |s1|
124         s1.sens_anot }
125     }.flatten.size -%>

129 <h3> sprawdzenie weryfikacji </h3>
130 <ul>
131     <% if 0==1 and UNatrora.x
132         ile=0
133         Akapit.zaanotowane.each do |z|
134             akaty=z.akapit_transzy
135             akat=akaty[0]
136             akat1=akaty[1]
137             w = akat.weryfikacja
138             unless w[:werdykt] == 1
139                 s = w[:sensy].collect{ |se|
140                     se0=SensAnot.znajdz(se, akat.akat_hash)
141                     if se0
142                         se0=se0.sensy_leksemu.nowy.to_s
143                     else
144                         se0='nil'
145                     end
146                     se1=SensAnot.znajdz( se, akat1.akat_hash )
147                     if se1
148                         se1=se1.sensy_leksemu.nowy.to_s
149                     else
150                         se1='nil'
151                     end
152                     Token.find(se).orth + ":" + se0 + '/' + se1
153                 }
154                 ile+=1
155             -%>
156 <%= "<li> #{akat.akapit_id}: #{akat.akapit.tresc} <br/>
157     dyzambiguacje: #{w[:disas].join(', ')} <br/> sensy:
158     #{s.join(', ')} <br/> frazy: #{w[:frazy].join(', ')}
159     </li>" -%>
160 <% end end end -%>
161 </ul>
162     Zdań niezgodnych po weryfikacji: <%= ile -%>

163 </div>

166 <h3> Tajemnica ginących i podwójnych sensów </h3>

168 <% ls = Leksem.find(:all, :include => :sensy_leksemu)

```

```

169   ls2= ls.collect{ |l|
170     s=l.sensy_leksemu.collect{|s| s.l.sensy.sens_ozn}
171     if s==s.uniq
172       nil
173     else
174       l.lemat
175     end
176   }.compact
177   -%>

179   <p> Wszystkich leksemów: <%= ls.size %>
180   <br/>
181   leksemów z podwójnym sensem: <%= ls2.size %>
182   <br/>
183   <%= ls2.join(', ') %>

188   <h3> sprawdzenie weryfikacyj </h3>
189   <ul>
190     <%
191       ile=0
192       Zdanie.zaanotowane.each do |z|
193         zt=z.zdania_transzy[0]
194         w = zt.weryfikacja
195         unless w[:werdykt] == 1
196           s = w[:sensy].collect{ |se| Token.find(se).orth}
197           ile+=1
198         -%>
199         <%= "<li> #{zt.zdanie_id}: #{zt.zdanie.tresc}      <br/> dyzambiguacje:
200           #{w[:disas].join(', ')}      <br/> sensy: #{s.join(', ')}      <br/>
201           frazy: #{w[:frazy].join(', ')}      </li>" -%>
202         <% end end -%>
203       </ul>
204       Zdań niezgodnych po weryfikacji: <%= ile -%>

208   <h3> klasy gram. </h3>
209   <% KlasaGram.kgs.each_pair do |k, v| -%>
210   <%= "#{k}: [{#{v.join(', ')}]}"
211   -%>

215   <% end -%>

219   <h3> Anotatorzy i status trefnych zdań </h3>

221   <% [ 369, 411, 1528, 4473, 4874, 5062, 5801].each do |zi|
222     z=Zdanie.find(zi)
223     z.zdania_transzy.each do |zt|

225       ui=zt.get_uzid
226       if ui
227         u = Uzytkownik.find(ui)
228         lo = u.login
229       else lo= "(nil)"
230     end

```

```

232     -%>

234     <%= "#{zt.zdanie_id}, #{zt.transza_auto.opis}, #{lo}:
      #{zt.status_tekst}" -%>

235     <br/>
236     <% end end -%>

240     <h3> Ile zaanotowali poszczególni Anotatorzy: </h3>

242     <% zest = Hash.new(0)
243         ZdaniaTranszy.find(:all).each { |zt|
244             if zt.ma_status?( :>, :niezatwierdzone )
245                 zest[zt.get_uzid] += 1
246             end
247         }
248         zest.each do |key, val|
249             -%>
250             <p> <%= Uzytkownik.find(key).login + ": " + val.to_s -%>
251             </p>
252             <% end -%>

255     <h3> Przypisanie transz </h3>
256     <ul>
257         <% TranszaAuto.find(:all).each do |tra| -%>
258         <li>
259             <%= "#{tra.opis}: #{if tra.uzytownik: tra.uzytownik.login ; else
                " end}" -%>
260         </li>
261         <% end -%>
262     </ul>

264     <div> ----- </div>

267     <% Zdanie.find(25).token.each { |tok|
268         if tok.do_sensu?(4)
269             ts = tok.sens(4)
270             if ts
271                 -%>
272                 <%= ts.sensy.sens_ozn %><br/>
274             <% end end } -%>

278     <%
279         les = Leksem.find(:all,
280             :include => :klasa_gram,
281             :conditions =>
282                 " klasa_gram.klasa_gram_ozn in ('ppron12', 'ppron3', 'siebie') "
283             )
284         les.each do |le|
285             %>
286             <p>
287                 <%= "#{le.lemat} #{le.klasa_gram.klasa_gram_ozn}"
288             %>
289             </p>

```

```

291   <% end %>

294   <% FrazaTyp.typy_glow.sort.each do |tg| -%>
295   <%= tg[0] + ": " + tg[1].join(", ") %><br/>
296   <% end %>

299   <% FrazaTyp.find(:all).each do |ft| %>
300   <p> <%= ft.typ_symbol %>
301   <br/> synhead:
302   /<%= ft.synhead_list %>/
303   <% ft.klasy_glowy(:synh).each {|kg1| %>
304   /<%= kg1.join(", ") %>/ <br/>
305   <% } %>
306   semhead: /<%= ft.semhead_list %>/
307   <% ft.klasy_glowy(:semh).each {|kg1| %>
308   /<%= kg1.join(", ") %>/ <br/>
309   <% } %>

311   </p>

313   <% end %>

```

### debug.rhtml

```

24   <div id="main">
25   <h1>nieoficjalna stronka do sprawdzania różnych rzeczy przez
      natrora</h1>

27   <p> z bazy danych:
28   <%= Komentarz.find_by_sql( "select current_timestamp" )[0].inspect %>
29   </p>

31   <%= link_to "Add a dummy" , :action => :create_dummies -%>

34   <p> z Railsów: <%= Time.now %></p>

36   <h2> Dummy1 </h2>
37   <ul>
38   <% Dummy1.find_all.each { |d| %>
39   <li>
40   <%= d.inspect %>
41   </li>
42   <% } %>
43   </ul>

47   <%
48   ## # Local Variables:
49   ## # mode: mmm
50   ## # End:
51   -%>

```

### elektroforeza.rhtml

```

24   <div id="main">
25   <h1>stronka (debug/elektroforeza) wiadoma wyłącznie AP, ŁD i GM,<br/>

```



**app/views/layouts****\_bug\_reports.rhtml**

```

24  <p style="font-size: xx-small">
25  Uwagi dotyczące działania serwisu i&nbsp;ewentualne błędy należy
26  zgłaszać na trackerze.
27  <br/>    W przypadku błędu proszę podać możliwie dokładną godzinę
28  jego        wystąpienia i/lub załączyć błąz Anotatorni.
29  </p>

```

**\_notice.rhtml**

```

24  <% if flash[:notice] -%>
25  <p style="padding-top: 6em;"&nbsp;</p>
26  <div id="notice"><a name="notizia"></a>
27  <%= flash[:notice] %></div>
28  <% end -%>

```

**\_powitalny.rhtml**

```

24  <h1>Anotatornia pozdrowienie Ci śle, <b><%=
25  Uzytkownik.find(session[:uzid]).login
26  %></b>.</h1>
27  <p>Jest <%= Time.now.to_s(:db) %>. <br/>
28  <%= "[" + PoziomyAnotacji.działające_nom.join(", ") + "]"<br/>" if
    DlaEli.nk -%>

30  Mamy <%= c194 = AkapitTranszy.count_diai %> akapit/y/ów,
31  w tym <%= AkapitTranszy.ile_zaanotowanych %>
32  zweryfikowany/e/ych<% if Rola.anotator?( session[:rola_id] ) %>,
33  <br><b> w tym
34  <%= AkapitTranszy.ile_zaanotowanych( session[:uzid] ) %>
35  przez Ciebie</b><% end %>.

37  <% if r_wszewid? %>
38  <br/>
39  Jak dotąd,
40  pobrano
41  <%= p194 = Akapit.ile_pobrzanych %>
42  -
43  <%
44  zost194 = c194 - p194
45  zostext194 = "#{zost194} (#{zost194 / AnoVersion.rozmiar_transzy * 2}
    transz/a/e)"
46  if Statusy.count( :conditions => "word_senses < 14" ) < 666 and
    zost194 <= 399 %>
47  <span style="background-color: #ff0;">
48  <b>
49  do pobrania już tylko <%= zostext194 %>!</b></span>

```

```

50      <% else # do pobrania jeszcze dużo
51      %>
52      do pobrania jeszcze <%= zostext194 %>.
53      <% end # of wyróżniać czy nie wyróżniać ile jeszcze do pobrania
54      end # of if wszewid or not
55      -%>
56      </p>

58      <%
59      ## # Local Variables:
60      ## # mode: mmm
61      ## # End:
62      -%>

```

### \_sidebar1.rhtml

```

24      <% # to jest partial używany przez szablon i szablon admin, a
25      # także zarządca
26      -%>

28      <div id="side" >
29      <% if session[ :uzid ] %>

31      <b> <%= session[ :login ] %></b>
32      (<%=
33      if session[ :drag_queen_id ]
34      "DQ jako "
35      end
36      %><%= Rola.nazwa_rol( session[ :rola_id ] )%>)
37      &nbsp; Anotatornia <b><%= AnoVersion.text %></b>
38      &nbsp;
39      <% if Rola.anotator?(session[:rola_id]) -%>
40      <%= link_to "Zweryfikowane" , :controller =>
41      annotation_controller, # zdef. w application_helper.rb
42      :action => 'ogladactwo' -%>
43      <%= link_to "Transze" , :controller => annotation_controller,
44      :action => 'lista_transz'
45      -%>

46      <% elsif not Rola.admin?(session[:rola_id]) # to znaczy dla gościa,
47      audytora i zarządcy
48      -%>

49      <%= link_to "Strona główna" , :controller =>
50      annotation_controller, :action => 'ogladactwo' -%>

51      <% if Rola.wszewid?(session[:rola_id]) -%>
52      <%= link_to "Zweryfikowane" , :controller =>
53      annotation_controller, :action => 'zweryf' -%>
54      <% end -%>

55      <% end # of if Rola.anotator? elsif not Rola.admin?
56      -%>

```



```

58     <% if session[:transza] %>
59     <%= link_to "Bieżąca transza" , :controller =>
        annotation_controller,
60     :action => 'pokaz_transze', :id => session[:transza]
61     -%>
62     <% end -%>

64     <% if DlaEli.x and not Rola.admin?(session[:rola_id]) -%>
65     <%= link_to "Wyszukiwanie" , :controller =>
        annotation_controller,
66     :action => 'wyszukaj_akapity'
67     -%>

69     <%= if session[:szukaj_akapity_transzy]
70     link_to "Bieżące wyszukiwanie" , :controller => annotation_controller,
        :action => 'wyszukaj_akapity'
71     end -%>
72     <% end -%>

74     <% unless r_admin? -%>
75     <%= link_to "Sł. sensów" , :controller => 'anotacja',
76     :action => 'sense_inventory'
77     -%>
78     <% end -%>

80     <% if Rola.admin?(session[:rola_id]) -%>
81     <%= link_to "Lista użytkowników" , :controller => 'admin',
        :action => 'lista_uzytkownikow' -%>

83     <%= link_to "Dodaj użytkownika" , :controller => 'admin', :action
        => 'dodaj_uzytkownika' -%>
84     <% else -%>
85     <%= link_to "Zmiana hasła" , :controller => 'admin',
86     :action => 'zmien_haslo', :id => session[:uzid] -%>
87     <% end -%>

89     <%= link_to "Wyloguj" , :controller => 'logowanie', :action =>
        'wylogowanie' -%>

91     <% end # of if session[:uzid]
92     -%>
93     </div>

97     <%
98     ## # Local Variables:
99     ## # mode: mmm
100    ## # End:
101    -%>

```

**admin.rhtml**

```

24    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
25    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
26    <html>

```

```

27     <head>
28       <title>Anotatornia</title>
29       <%= javascript_include_tag :defaults # 'prototype', 'effects',
        'dragdrop'
30       %>
31       <%= stylesheet_link_tag "scaffold" , "application", # "admin" ,
32       :media => "all" %>
33     </head>
34     <body id="admin">

37     <% if session[:uzid] -%>
38     <div id="expiry"></div>
39     <%= # periodically_call_remote :url => :controller => 'logowanie',
40     # :action => 'session_expiry', \unskip
41     # :frequency => 15
42     -%>
43     <% end -%>

45     <!-- <div id="banner" > -->
46     <div id="columns" >
47     <%= render(:partial => "/layouts/sidebar1") %>

49     <%= yield :layout %>

51     <% if session[:uzid] -%>
52     <div class="ciemne-tlo">
53     <%= render :partial => '/layouts/bug_reports' %>
54     <div>
55     <% end -%>
56     </div>
57     </body>
58     </html>

```

### zdanie.rhtml

```

24     <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
25     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

27     <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
28     <head>
29     <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
30     <title>Anotatornia &#8212; Zdanie: <%= controller.action_name
        %></title>
31     <%= stylesheet_link_tag 'application' %>
32     </head>
33     <body>

35     <div id="columns" >
36     <%= render (:partial => "/layouts/sidebar1") %>

38     <div id="main" >
39     <p style="color: green"><%= flash[:notice] %></p>

41     <%= yield %>

```

```

42     </div>
44 </body>
45 </html>

```

## app/views/logowanie

### \_session\_expiry.rhtml

```

24 <span style="color: #bfb; background: #404; font-weight: bold">
25   Twoja sesja wygaśnie za <%= session_expiry %> sekund/y
26 </span>

```

### dodaj\_uzytkownika.rhtml

```

24 <h1>Logowanie#dodaj_uzytkownika</h1>
25 <p>(Jestem w: app/views/logowanie/dodaj_uzytkownika.rhtml)</p>
26 <div class="depot-form" >
27   <%= error_messages_for 'uzytkownik' %>
28   <fieldset>
29     <legend>Wprowadź dane użytkownika</legend>
30     <% form_for :uzytkownik do |form| %>
31       <p>
32         <label for="uzytkownik_rola_id" >Rola:</label>
33         <br/>
34         <%=
35           form.select :rola_id,
36             (Rola.find(:all, :order => :rola_id)).collect {|rola|
37               [rola[:opis], rola[:rola_id]]
38             },
39           :prompt => "Wskaż rolę użytkownika"
40         %>
41       </p>
42       <p>
43         <label for="uzytkownik_login" >Login:</label>
44         <br/>
45         <%= form.text_field :login, :size => 40 %>
46       </p>
47       <p>
48         <label for="uzytkownik_haslo" >Hasło:</label>
49         <br/>
50         <%= form.password_field :haslo, :size => 40 %>
51       </p>
52       <p>
53         <label for="uzytkownik_haslo_confirmation" >Powtórz
54           hasło:</label>
55         <br/>
56         <%= form.password_field :haslo_confirmation, :size => 40 %>
57       </p>

```

```

58     <label for="uzytkownik_imie" >Imię użytkownika:</label>
59     <br/>
60     <%= form.text_field :imie, :size => 40 %>
61     </p>
62     <p>
63         <label for="uzytkownik_nazwisko" >Nazwisko użytkownika:</label>
64         <br/>
65         <%= form.text_field :nazwisko, :size => 40 %>
66     </p>
67     <p>
68         <label for="uzytkownik_email" >email użytkownika:</label>
69         <br/>
70         <%= form.text_field :email, :size => 40 %>
71     </p>
72     <%= submit_tag "Dodaj użytkownika" , :class => "submit" %>
73     <% end %>
74 </fieldset>
75 </div>

```

### index.rhtml

```

24 <div id="main">
25 <%= render :partial => '/layouts/notice' -%>
26 <%= render :partial => '/layouts/powitalny' -%>
27 </div>

```

### lista\_uzytkownikow.rhtml

```

24 <h1>Lista użytkowników</h1>
25 <ul>
26   <% for uzytkownik in @wszyscy_uzytkownicy %>
27     <li><%= link_to "[X]", { # link_to options
28       :controller => 'logowanie',
29       :action => 'usun_uzytkownika',
30       :id => uzytkownik},
31     { # html options
32       :method => :post,
33       :confirm => "Czy rzeczywiście usunąć #{uzytkownik.login}?"
34   } %>
35     <%= h(uzytkownik.login) + " (" + uzytkownik.rola.opis_kr+ ")" %>
36   </li>
37   <% end %>
38 </ul>

```

### logowanie.rhtml

```

24 <div id="main">
25 <%= render :partial => '/layouts/notice' -%>
27 <h1>Anotatornia
28 <%= AnoVersion.text %>
29 &#0151; logowanie</h1>

```

```

30  <div class="depot-form" >
31    <fieldset>
32      <legend>Zaloguj się</legend>
33      <% form_tag do %>
34        <p>
35          <label for="login" >Login:</label>
36          <%= text_field_tag :login, params[:login] %>
37        </p>
38        <p>
39          <label for="haslo" >Hasło:</label>
40          <%= password_field_tag :haslo, params[:haslo] %>
41        </p>
42        <p>
43          <%= submit_tag "Zaloguj" %>
44        </p>
45      <% end %>
46    </fieldset>
47  </div>
48  <br/>
49
50  
51  <span style="font-size: xx-small">crafted in Ruby on Rails by natror
52    at o2.pl</span>
53  <br/>
54  
55
56  </div>

```

### usun\_uzytkownika.rhtml

```

24  <h1>Logowanie#usun_uzytkownika</h1>
25  <p>Find me in app/views/logowanie/usun_uzytkownika.rhtml</p>

```

### wylogowanie.rhtml

```

24  <h1>Wylogowanie</h1>
25  <p>Jestem w: app/views/logowanie/wylogowanie.rhtml</p>

```

### zablokuj\_uzytkownika.rhtml

```

24  <h1>Blokowanie użytkownika</h1>
25  <ul>
26    <% uzytkownik = Uzytkownik.find(params[:id]) %>
27    <li><%= link_to "[zablokuj]" , { # link_to options
28      :controller => 'logowanie',
29      :action => 'zablokuj_uzytkownika',
30      :id => uzytkownik},
31      { # html options
32      :method => :post,
33      :confirm => "Czy rzeczywiście zablokować #{uzytkownik.login}?"
34    } %>
35    <%= h(uzytkownik.login) + " (" + uzytkownik.rola.opis_kr+ ")" %>

```

```

36     </li>
37 </ul>

```

## app/views/zarzadca

### lista\_anotatorow.rhtml

```

24 <div id="main">
25 <%= render :partial => '/layouts/notice' -%>

27 <h1>Zarządca &#8212; lista anotatorów</h1>
28 <ul>
29   <% for anotator in @wszyscy_anotatorzy %>
30     <li>

32       <%= h(anotator.login) %>
33       ma przypisane:
34       <ul>
35         <% anotator.transza_auto.each do |tra| %>
36           <li><%= tra.opis %>

37 <%=
38   link_to "[odbierz]" , { # link_to options
39     :controller => 'zarzadca',
40     :action => 'odbierz_transze',
41     :id => tra.transza_auto_id,
42     :anotator => anotator.uzytownik_id},
43     { # html options
44     :method => :get }
45   %>
46     </li>
47   <% end %>
48   </ul>
49 </li>
50 <% end %>
51 </ul>

53 </div>

```

### odbierz\_transze.rhtml

```

24 <div id="main">
25 <%= render :partial => '/layouts/notice' -%>

27 <h1>Zarządca &#8212; odbierz transzę</h1>

29 <p> Zamierzasz odebrać transzę
30 <%= "#{@transza.transza_auto_id}:
   \#{@transza.czasownik_auto.bezokolicznik}\" %>
31 <br/>
32 <%= "nie-" unless Rola.anotator?(@anotator.rola_id) %>anotatorowi
33 <%= @anotator.login %>
34 </p>

```

```

37  <%=
38    button_to "odbierz" , { # link_to options
39      :controller => 'zarzadca',
40      :action => 'odbierz_transze',
41      :id => @transza.transza_auto_id,
42      :anotator => @anotator.uzytkownik_id},
43    { # html options
44      :method => :post,
45      :confirm => "Czy rzeczywiście odebrać
46        transzę#{@transza.transza_auto_id}:
47        \#{@transza.czasownik_auto.bezokolicznik}\#{@anotatorowi
48        #{@anotator.login}?"
49    }
50  %>
51 </div>

```

## przypisz\_transze.rhtml

```

24 <h1>Zarzadca#przypisz_transze</h1>
25 <p>Find me in app/views/zarzadca/przypisz_transze.rhtml</p>

```

## db/migrate

### 001\_add\_updated.rb

```

23 class AddUpdated < ActiveRecord::Migration
24   def self.up
25     bloobraZ = false
26
27     adup = Proc.new { |model|
28       unless model.column_names.include?( "updated_at" )
29         execute "alter table #{model.table_name} add column updated_at
30           timestamp"
31         bloobraZ = true
32       end
33     }
34
35     adup.call(CzM)
36     adup.call(FrazaTyp)
37     adup.call(MorphosyntacticRozbieznosc)
38     adup.call(Leksem)
39     adup.call(Komentarz)
40     adup.call(KlasaGram)
41     adup.call(Protokol)
42
43     unless PunktProtokolu.column_names.include?("created_at") and
44       PunktProtokolu.column_names.include?("updated_at")
45       execute "alter table punkt_protokolu rename to pp_old;"
46       execute "CREATE TABLE IF NOT EXISTS punkt_protokolu(
47         punkt_protokolu_id integer primary key autoincrement,
48         elt_protokol_id integer not null,          elt_protokol_type text

```

```

    not null,          protokol_id integer not null,
    akapit_transzy_id integer not null,          uzytkownik_id
    integer,created_at timestamp default
    current_timestamp,updated_at timestamp);"
47 execute "insert into punkt_protokolu(          punkt_protokolu_id,
    elt_protokol_id,          elt_protokol_type,
    protokol_id,akapit_transzy_id,          uzytkownik_id)select
    punkt_protokolu_id,          elt_protokol_id,
    elt_protokol_type,          protokol_id,akapit_transzy_id,
    uzytkownik_id from pp_old;"
48 execute "drop table pp_old;"

    bloobraZ = true
51 end # of unless PunktProtokolu...

53 adup.call(Rola)
54 adup.call(SgVariant)
55 adup.call(KoniecZdaniaAnot)
56 adup.call(SentencesRozbieznosc)
57 adup.call(Path)
58 adup.call(SegmentationRozbieznosc)
59 adup.call(Sensy)
60 adup.call(Tagset)
61 adup.call(Transza)
62 adup.call(Akapit)
63 adup.call(Morph)

65 execute "vacuum" if bloobraZ
66 end

68 def self.down
69 end
70 end

```

## 002\_komentarz\_add\_nowy.rb

```

23 class KomentarzAddNowy < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     sth_added = self.addcolumn( 'komentarz', 'nowy', "boolean default
    't'")
29
30     execute "vacuum" if sth_added
31
32   end
33
34   def self.down
35     end
36 end

```



**003\_komentarz\_index\_created.rb**

```

23  class KomentarzIndexCreated < ActiveRecord::Migration
24    def self.up
25      execute "CREATE INDEX IF NOT EXISTS komentarz_created_at
           on komentarz(created_at)"
26    end
27
28    def self.down
29      end
30  end

```

**004\_token\_add\_kolejnosc.rb**

```

23  class TokenAddKolejnosc < ActiveRecord::Migration
24
25    extend MigrationHelper
26
27    def self.up
28      sth_added = addcolumn( 'token', 'kolejnosc', 'integer' )
29
30      unless Token.column_names.include?( "kolejnosc" )
31        execute "alter table token rename to t0"
32
33      execute "CREATE TABLE IF NOT EXISTS token(  token_id integer primary
key autoincrement,  kolejnosc integer not null,  akapit_id integer
not null,  xpointer text,  segmentation_xmlid text,
morphosyntactic_xmlid text,  fs_morph_comment text,  path_id
integer not null,  orth text,  czy_interp boolean not null,
ns_poprzedza boolean default 'f',  ns_nastepuje boolean,
czy_konczy_zdanie boolean,  czy_konczy_zdanie_updated_at timestamp,
sg_choice_id integer null default null,  sg_variant_id integer null
default null,  dodany boolean default 'f' not null,  chosen boolean
default 't', - dedukowalne z sg_variant_anot  chosen_updated_at
timestamp,  created_at timestamp default current_timestamp,
updated_at timestamp default current_timestamp)"
34
35      execute " insert into token(  token_id,  kolejnosc,  akapit_id,
xpointer,  segmentation_xmlid,  morphosyntactic_xmlid,
fs_morph_comment,  path_id,  orth,  czy_interp,  ns_poprzedza,
ns_nastepuje,  czy_konczy_zdanie,  czy_konczy_zdanie_updated_at,
sg_choice_id,  sg_variant_id,  chosen,  chosen_updated_at,
created_at,  updated_at)select  token_id,token_id * 216 as
kolejnosc,  akapit_id,  xpointer,  segmentation_xmlid,
morphosyntactic_xmlid,  fs_morph_comment,  path_id,  orth,
czy_interp,  ns_poprzedza,  ns_nastepuje,  czy_konczy_zdanie,
czy_konczy_zdanie_updated_at,  sg_choice_id,  sg_variant_id,
chosen,  chosen_updated_at,  created_at,  updated_atfrom t0"
36
37      execute "drop table t0"
38      sth_added = true
39    end
40  end

```

```

42     unless Token.find_by_sql( "select * from token where kolejnosc is
43       not null limit 1" )[0]
44       execute "update token set kolejnosc = token_id*216"
45     end
46
47     addindex( 'token', 'akapit_id' )
48     addindex "token", "czy_interp"
49     addindex "token", 'ns_nastepuje'
50     addindex 'token', 'ns_poprzedza'
51     addindex 'token', [ 'path_id', 'morphosyntactic_xmlid' ]
52     addindex 'token', [ :path_id, :segmentation_xmlid ], :unique
53     addindex :token, :sg_choice_id
54     addindex :token, :sg_variant_id
55
56     addindex( 'token', 'kolejnosc', :unique )
57     addindex( 'token', 'dodany' )
58
59     execute "vacuum" if sth_added
60   end
61
62   def self.down
63   end
64 end

```

#### **006\_sg\_choice\_variant\_add\_dodany.rb**

```

23 class SgChoiceVariantAddDodany < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     addcolumn :sg_choice, :dodany, "boolean default 'f'"
29     addcolumn :sg_variant, :dodany, "boolean default 'f'"
30   end
31
32   def self.down
33   end
34 end

```

#### **007\_statusy\_add\_akapit\_id.rb**

```

23 class StatusyAddAkapitId < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     sth_added = addcolumn( 'statusy', 'akapit_id', 'integer' )
29     if sth_added
30       execute "update statusy set akapit_id=( " +
31         " select akapit_id from akapit_transzy at " +
32         " where at.akapit_transzy_id = statusy.akapit_transzy_id) " +
33         " where akapit_id is null"
34     end # of if sth. added
35   end
36
37   def self.down
38
39   end
40 end

```

```

39     end
40 end

```

### oo8\_token\_add\_superancje.rb

```

23 class TokenAddSuperancje < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     addcolumn( 'token', 'superancja', 'text' )
29     addcolumn( 'protokol', 'czy_superancja', 'boolean' )
30   end
31
32   def self.down
33     end
34 end

```

### oo9\_spatialize\_npses.rb

```

23 class SpatializeNpses < ActiveRecord::Migration
24   def self.up
25     aks = Akapit.find_by_sql( "select * from akapit inner join path
26                               using( path_id ) where path_text like '%310-1-000001%' or path_text
27                               like '%330-1-000003%' or path_text like '%330-1-000004%';")
28     aids= aks.collect{ |a| a.id }
29     ts = Token.find( :all, :conditions => { :akapit_id => aids,
30       :ns_poprzedza => true, :czy_interp => false } )
31     ts.each{ |t|
32       if %w(a i o u w z się jedynie A I O U W Z Się Jedynie).include?(
33         t.poprzedni.orth )
34         t.ns_poprzedza = false
35         t.save!
36       end
37     }
38
39     Token.update_ns_nastepuje( aids.min, aids.max, :force ) if aids[0]
40
41   end
42
43   def self.down
44     end
45 end

```

### o10\_reverse\_niedostrzeganiem\_masturbacji.rb

```

23 class ReverseNiedostrzeganiemMasturbacji < ActiveRecord::Migration
24   # Ma sens tylko w wersji 8003, gdzie niefortunnie dodałem wariant
25   # segmentacyjny „niedostrzeganiem” w akapicie 19.
26
27   def self.up
28     if AnoVersion.nkjp?
29       if ( a = Akapit.find( :first, :conditions => "akapit_id=19" )
30         # Akapit.find( 19 ) causes error if absent

```

```

30      ) and a.tresc =~ /\^0ddramatyzowanie problemu masturbacji nie
      może też łączyć się.* nie dostrzeganiem niedostrzeganiem/
31      t = Token.find( :first, :conditions =>{ :orth =>
        'niedostrzeganiem', :akapit_id => 19 })
32      if t
33        sgc_id = t.sg_choice_id
34        t.sg_choice.destroy
35        t.destroy
36        execute "update token set chosen='t', sg_variant_id=null,
          sg_choice_id=null where sg_choice_id=#{sgc_id}"
37        # Akapit.dopuszcz_posegmentowane( 19, 19 ) nie ma sensu, bo tam
          jest wariant segmentacyjny
38      end
39    end
40  end
41 end

43 def self.down
44   end
45 end

```

### 011\_add\_sensowe.rb

```

23 class AddSensowe < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     sth_done =
29       addcolumn 'interpretacja', 'cz_m_leksem_id', 'integer'
30       addcolumn 'interpretacja', 'sensy_id', 'integer'
31
32       addcolumn 'leksem', 'cz_m_leksem_id', 'integer'
33
34     unless Interpretacja.find( :first, :conditions => "sensy_id is not
35       null" ) or
36       ( tables.include?( "sens_annot" ) and SensAnot.find( :first ) )
37       droptable :cz_m
38       droptable :sensy
39       droptable :sens_annot
40       droptable :cz_m_leksem
41     end
42
43     execute "CREATE TABLE IF NOT EXISTS cz_m(- Attributes -cz_m_id
44       integer primary key autoincrement,cz_m_ozn text,created_at
45       timestamp,updated_at timestamp);"
46
47     execute "CREATE TABLE IF NOT EXISTS sensy(- Attributes -
48       sensy_id integer primary key autoincrement, cz_m_id integer not
49       null, cz_m_leksem_id integer not null, xmlid text, n
50       integer, - przy wlewie sprawdzamy czy lemat.odpolszcz.n == xmlid
51       short_def text, long_def_xml text, long_def_html text,
52       created_at timestamp, updated_at timestamp);"
53   end
54 end

```

```

47     addindex :sensy, :cz_m_id
48     addindex :sensy, :xmlid, :unique
50     execute "CREATE TABLE IF NOT EXISTS sens_anot(- Attributes -
      sens_anot_id integer primary key autoincrement,  uzytkownik_id
      integer,  akapit_transzy_id integer not null,  interpretacja_id
      integer not null,  sensy_id integer not null,  automatycznie
      boolean,  created_at timestamp,  updated_at timestamp);"
52     addindex :sens_anot, :sensy_id
53     addindex :sens_anot, :interpretacja_id
54     addindex :sens_anot, [:interpretacja_id, :akapit_transzy_id]
55     addindex :sens_anot, :akapit_transzy_id
57     execute "CREATE TABLE IF NOT EXISTS cz_m_leksem(- Attributes
      -cz_m_leksem_id integer primary key autoincrement, lemat text not
      null,xmlid text not null,cz_m_id integer not null,created_at
      timestamp,updated_at timestamp);"
59     addindex :cz_m_leksem, [:lemat, :cz_m_id], :unique
60     addindex :cz_m_leksem, :xmlid, :unique
62     execute "vacuum;"
64     # zainicjować cz_m
65     if sth_done
66       Tagset.zainicjuj( :force )
67       KlasaGram.zsynchronizuj_z_tagsetem
68       CzM.zainicjuj
69     end
71   end
73   def self.down
74     end
75   end

```

## 012\_nowa\_segmentacja\_add\_nps.rb

```

23   class NowaSegmentacjaAddNps < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28       addcolumn 'nowa_segmentacja', 'nps', "boolean not null default
        't'"
30       dropindex :nowa_segmentacja_seg_is
31       dropindex :nowa_segmentacja_seg_is_ids
32       dropindex :nowa_segmentacja_seg_is_ids_nps
33       addindex :nowa_segmentacja, [:seg_is, :seg_should_be, :ids, :nps
        ], :unique
35     end
37     def self.down
38       end
39   end

```

**013\_dummy\_table.rb**

```

23  class DummyTable < ActiveRecord::Migration
24    def self.up
25
26      if AnoVersion.port == 8004
27
28        execute "create table if not exists dummy1(dummy1_id integer
                primary key autoincrement,tresc text,created_at timestamp);"
29
30        execute "create table if not exists dummy2(dummy2_id integer
                primary key autoincrement,tresc text,created_at timestamp
                default current_timestamp);"
31
32      end # of if 8004
33
34    end # of up
35
36    def self.down
37      end
38    end

```

**014\_przeklnij\_morphosynty.rb**

```

23  class PrzeklnijMorphosynty < ActiveRecord::Migration
24    def self.up
25      if AnoVersion.nkjp?
26        { :sentences=>[1048, 1049, 3004, 3310, 5097, 5502, 5893, 5894,
                        6380, 6386, 7853, 7865, 7881, 7905], :morphosyntactic=>[4, 26,
                        28, 43, 78, 98, 102, 122, 134, 137, 176, 184, 206, 219, 314, 315,
                        712, 723, 731, 737, 750, 758, 1018, 1019, 1020, 1021, 1022, 1023,
                        1024, 1025, 1026, 1027, 1028, 1032, 1033, 1035, 1036, 1037, 1038,
                        1039, 1040, 1041, 1042, 1044, 1045, 1046, 1047, 1050, 1051, 1052,
                        1053, 1054, 1055, 1056, 1057, 1058, 1059, 1061, 1062, 1063, 1064,
                        1065, 1066, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1078,
                        1079, 1080, 1081, 1082, 1083, 1085, 1086, 1088, 1089, 1091, 1092,
                        1093, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1388, 1398, 1399,
                        1416, 1482, 1497, 1744, 1745, 1759, 1760, 1761, 1776, 1777, 1785,
                        1800, 1801, 1802, 1807, 1809, 1816, 1821, 1823, 1833, 1865, 1880,
                        1884, 1885, 1886, 1889, 1909, 1915, 1923, 1926, 1933, 1935, 1941,
                        1954, 1973, 1975, 1989, 2013, 2014, 2023, 2063, 2079, 2084, 2086,
                        2089, 2443, 2444, 2449, 2498, 2526, 2573, 2578, 2590, 2604, 2690,
                        2707, 2711, 2783, 2788, 2815, 2834, 2840, 2844, 2853, 2857, 2875,
                        2914, 2923, 2930, 2933, 2936, 2937, 2938, 2941, 2949, 2965, 2979,
                        2993, 3029, 3054, 3059, 3062, 3118, 3201, 3261, 3269, 3281, 3284,
                        3286, 3299, 3301, 3306, 3338, 3368, 3374, 3377, 3394, 3441, 3447,
                        3464, 3470, 3473, 3489, 3560, 3564, 3578, 3580, 3602, 3603, 3614,
                        3620, 3622, 3627, 3633, 3638, 3639, 3666, 3713, 3749, 3750, 3752,
                        3759, 3770, 3785, 3792, 3803, 3826, 3843, 3850, 3852, 3877, 3886,
                        3910, 3944, 3945, 3950, 3951, 4001, 4019, 4026, 4028, 4043, 4081,
                        4082, 4107, 4115, 4116, 4138, 4152, 4165, 4180, 4248, 4286, 4288,

```

```

4309, 4316, 4357, 4374, 4386, 4387, 4402, 4408, 4435, 4437, 4457,
4550, 4566, 4643, 4644, 4645, 4753, 4953, 4957, 4960, 4971, 4978,
4987, 5009, 5016, 5038, 5040, 5042, 5064, 5127, 5128, 5137, 5140,
5191, 5193, 5202, 5203, 5216, 5267, 5269, 5290, 5293, 5303, 5312,
5362, 5375, 5390, 5399, 5410, 5456, 5466, 5548, 5556, 5564, 5572,
5573, 5615, 5682, 5683, 5727, 5747, 5752, 5774, 5789, 5833, 5836,
5854, 5861, 5862, 5943, 5985, 6019, 6024, 6076, 6110, 6203, 6224,
6265, 6302, 6320, 6401, 6431, 6442, 6468, 6478, 6500, 6531, 6548,
6552, 6573, 6603, 6638, 6650, 6654, 6663, 6664, 6666, 6681, 6690,
6711, 6722, 6726, 6730, 6750, 6775, 6837, 6850, 6884, 6936, 6966,
7003, 7009, 7058, 7120, 7131, 7133, 7180, 7200, 7228, 7258, 7334,
7349, 7361, 7368, 7373, 7375, 7390, 7429, 7447, 7449, 7498, 7505,
7556, 7591, 7598, 7610, 7625, 7648, 7655, 7656, 7663, 7700, 7702,
7703, 7705, 7736, 7775, 7791, 7794, 7978, 7981, 7983, 8025, 8033,
8054, 8056, 8090, 8179, 8181, 8187, 8220, 8238, 8248, 8434, 8841,
8884, 8903, 8913, 8916, 8922, 8997, 9007, 9678, 9689, 9691],
:segmentation=>[]}.
27     each_pair { |poz, ids|
28       Token.find( ids ).each { |t|
29         t.superancja ||=[]
30         t.superancja |= [ poz ]
31         t.save!
32       }
33     }

35     end # of if nkjp?
36     # z ręcznego przejrzenia było:
37     [4, 26, 28, 43, 78, 98, 102, 122, 134, 137, 176, 184, 206, 219, 283,
314, 315, 712, 723, 731, 737, 750, 758, 1388, 1398, 1399, 1416, 1482,
1497, 1744, 1745, 1759, 1760, 1761, 1776, 1777, 1785, 1800, 1801,
1802, 1807, 1809, 1816, 1821, 1823, 1833, 1865, 1880, 1884, 1885,
1886, 1889, 1909, 1915, 1923, 1926, 1933, 1935, 1941, 1954, 1973,
1975, 1989, 2013, 2014, 2023, 2063, 2079, 2084, 2086, 2089, 2443,
2444, 2449, 2498, 2526, 2573, 2578, 2590, 2604, 2690, 2707, 2711,
2783, 2788, 2815, 2840, 2844, 2853, 2857, 2923, 2930, 2933, 2936,
2937, 2938, 2941, 2949, 2965, 2979, 2993, 3029, 3054, 3059, 3062,
3118, 3261, 3269, 3281, 3284, 3286, 3299, 3301, 3306, 3338, 3368,
3374, 3377, 3464, 3470, 3473, 3489, 3560, 3564, 3578, 3580, 3602,
3603, 3614, 3620, 3622, 3627, 3633, 3638, 3639, 3666, 3713, 3749,
3750, 3752, 3759, 3770, 3785, 3792, 3803, 3826, 3843, 3850, 3852,
3877, 3886, 3910, 3944, 3945, 3950, 3951, 4001, 4019, 4026, 4028,
4043, 4081, 4082, 4107, 4115, 4116, 4138, 4152, 4165, 4180, 4248,
4286, 4288, 4386, 4387, 4402, 4408, 4953, 4957, 4960, 4971, 5682,
5683, 5727, 5747, 5752, 5774, 5789, 5833, 5836, 5854, 5861, 5862,
5943, 5985, 6019, 6024, 6500, 6531, 6548, 6552, 6711, 6722, 6726,
6730]

40     [1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028,
1032, 1033, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1044,
1045, 1046, 1047, 1050,

```

```

41      1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1061, 1062,
      1063, 1064, 1065, 1066, 1069, 1070, 1071, 1072, 1073, 1074, 1075,
      1076, 1078, 1079, 1080, 1081, 1082, 1083, 1085, 1086, 1088, 1089,
      1091, 1092, 1093, 1095, 1096, 1097, 1098, 1099, 1100, 1101]

43      # z samej ms było:
49      [4, 26, 28, 43, 78, 98, 102, 122, 134, 137, 176, 184, 206, 219, 314,
      315, 712, 723, 731, 737, 750, 758, 1018, 1019, 1020, 1021, 1022,
      1023, 1024, 1025, 1026, 1027, 1028, 1032, 1033, 1035, 1036, 1037,
      1038, 1039, 1040, 1041, 1042, 1044, 1045, 1046, 1047, 1050, 1051,
      1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1061, 1062, 1063,
      1064, 1065, 1066, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076,
      1078, 1079, 1080, 1081, 1082, 1083, 1085, 1086, 1088, 1089, 1091,
      1092, 1093, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1388, 1398,
      1399, 1416, 1482, 1497, 1744, 1745, 1759, 1760, 1761, 1776, 1777,
      1785, 1800, 1801, 1802, 1807, 1809, 1816, 1821, 1823, 1833, 1865,
      1880, 1884, 1885, 1886, 1889, 1909, 1915, 1923, 1926, 1933, 1935,
      1941, 1954, 1973, 1975, 1989, 2013, 2014, 2023, 2063, 2079, 2084,
      2086, 2089, 2443, 2444, 2449, 2498, 2526, 2573, 2578, 2590, 2604,
      2690, 2707, 2711, 2783, 2788, 2815, 2834, 2840, 2844, 2853, 2857,
      2875, 2914, 2923, 2930, 2933, 2936, 2937, 2938, 2941, 2949, 2965,
      2979, 2993, 3029, 3054, 3059, 3062, 3118, 3201, 3261, 3269, 3281,
      3284, 3286, 3299, 3301, 3306, 3338, 3368, 3374, 3377, 3394, 3441,
      3447, 3464, 3470, 3473, 3489, 3560, 3564, 3578, 3580, 3602, 3603,
      3614, 3620, 3622, 3627, 3633, 3638, 3639, 3666, 3713, 3749, 3750,
      3752, 3759, 3770, 3785, 3792, 3803, 3826, 3843, 3850, 3852, 3877,
      3886, 3910, 3944, 3945, 3950, 3951, 4001, 4019, 4026, 4028, 4043,
      4081, 4082, 4107, 4115, 4116, 4138, 4152, 4165, 4180, 4248, 4286,
      4288, 4309, 4316, 4357, 4374, 4386, 4387, 4402, 4408, 4435, 4437,
      4457, 4550, 4566, 4643, 4644, 4645, 4753, 4953, 4957, 4960, 4971,
      4978, 4987, 5009, 5016, 5038, 5040, 5042, 5064, 5127, 5128, 5137,
      5140, 5191, 5193, 5202, 5203, 5216, 5267, 5269, 5290, 5293, 5303,
      5312, 5362, 5375, 5390, 5399, 5410, 5456, 5466, 5548, 5556, 5564,
      5572, 5573, 5615, 5682, 5683, 5727, 5747, 5752, 5774, 5789, 5833,
      5836, 5854, 5861, 5862, 5943, 5985, 6019, 6024, 6076, 6110, 6203,
      6224, 6265, 6302, 6320, 6401, 6431, 6442, 6468, 6478, 6500, 6531,
      6548, 6552, 6573, 6603, 6638, 6650, 6654, 6663, 6664, 6666, 6681,
      6690, 6711, 6722, 6726, 6730, 6750, 6775, 6837, 6850, 6884, 6936,
      6966, 7003, 7009, 7058, 7120, 7131, 7133, 7180, 7200, 7228, 7258,
      7334, 7349, 7361, 7368, 7373, 7375, 7390, 7429, 7447, 7449, 7498,
      7505, 7556, 7591, 7598, 7610, 7625, 7648, 7655, 7656, 7663, 7700,
      7702, 7703, 7705, 7736, 7775, 7791, 7794, 7978, 7981, 7983, 8025,
      8033, 8054, 8056, 8090, 8179, 8181, 8187, 8220, 8238, 8248, 8434,
      8841, 8884, 8903, 8913, 8916, 8922, 8997, 9007, 9678, 9689, 9691]

51      true
53      end
55      def self.down
56          end
57      end

```



**015\_debug\_potworne\_czasy.rb**

```
23 class DebugPotworneCzasy < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     # addindex :interpretacja, :disamb już jest
29     addindex :interpretacja_anot, [ :token_id, :uzytkownik_id ]
30     addindex :interpretacja, [ :token_id, :disamb ]
31   end
32
33   def self.down
34     end
35   end
36 end
```

**016\_przyspiesz\_danowlew.rb**

```
23 class PrzyspieszDanowlew < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     addindex :leksem, [ :path_id, :lemat, :klasa_gram_id ], :unique
29     addindex :token, [ :akapit_id, :segmentation_xmlid, :orth ] # nie
30       unique, bo tylko do wyszukiwania przy danowlewie
31   end
32
33   def self.down
34     end
35   end
36 end
```

**017\_popraw\_lex\_xmlidy.rb**

```
23 class PoprawLexXmlidy < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     ido = Interpretacja.find( :all, :conditions => { :dodana => true } )
29     ido.each { |int|
30       if Interpretacja.find( :first, :conditions =>
31         [ " dodana = 'f' and token_id=? and lex_xmlid
32           = ? and leksem_id <> ? ",
33           int.token_id, int.lex_xmlid, int.leksem_id ]
34       )
35         puts int.created_at.to_s( :db )
36
37         nry_lt, lexids_h = int.zainnij
38
39         nr_lt = lexids_h[ int.leksem_id ]
40         nr_lt ||= nry_lt.max + 1
```

```

42     int.numer_lex_token = nr_lt
43     int.lex_xmlid = int.lex_xmlid.gsub( /(\d* )\.(\\d* )\.(\\d* )/,
      "\\1.\\2.#{nr_lt}" )
44     int.msd_xmlid = int.msd_xmlid.gsub( /(\d* )\.(\\d* )\.(\\d*
      )\.(\\d* )/, "\\1.\\2.#{nr_lt}.\\4" )
45     int.save!
46   end
47 }
48
49 end
50
51 def self.down
52   end
53 end

```

### 018\_przepisz\_baze.rb

```

24 class PrzepiszBaze < ActiveRecord::Migration
25
26   extend MigrationHelper
27
28   def self.up
29     if false
30       pragmas_for_update
31
32       przepisz_tabele "create TABLE IF NOT EXISTS cz_m(cz_m_id integer
        primary key autoincrement,cz_m_ozn text,created_at
        timestamp,updated_at timestamp);"
33
34       przepisz_tabele( "create TABLE IF NOT EXISTS sensy(  sensy_id
        integer primary key autoincrement,  cz_m_id integer not null,
        cz_m_leksem_id integer not null,  xmlid text,  short_def text,
        sensy_xml text,  created_at timestamp,  updated_at timestamp);",
35
36         "create index if not exists sensy_cz_m_id on sensy(
        cz_m_id );",
37
38         "create unique index if not exists sensy_xmlid on
        sensy( xmlid );")
39
40       przepisz_tabele( "create TABLE IF NOT EXISTS sens_anot(
        sens_anot_id integer primary key autoincrement,  uzytkownik_id
        integer,  akapit_transzy_id integer not null,  interpretacja_id
        integer not null,  sensy_id integer not null,  automatycznie
        boolean,  created_at timestamp,  updated_at timestamp);",
41
42         "create index IF NOT EXISTS sens_anot_sensy_id on
        sens_anot( sensy_id );",
43
44         " create index IF NOT EXISTS
        sens_anot_interpretacja_id on sens_anot(
        interpretacja_id );" ,
45
46         "create index IF NOT EXISTS
        sens_anot_interpretacja_id_akapit_transzy_id on
        sens_anot( interpretacja_id, akapit_transzy_id );",
47

```

```

49         "create index IF NOT EXISTS
           sens_anot_akapit_transzy_idon sens_anot(
           akapit_transzy_id );"
50     )
53     przepisz_tabele( "create TABLE IF NOT EXISTS
           cz_m_leksem(cz_m_leksem_id integer primary key autoincrement, lemat
           text not null, xmlid text not null, cz_m_id integer not
           null, created_at timestamp, updated_at timestamp);",
55         "create unique index IF NOT EXISTS
           cz_m_leksem_lemat_cz_m_idon cz_m_leksem (lemat,
           cz_m_id);",
57         "create unique index IF NOT EXISTS
           cz_m_leksem_xmlidon cz_m_leksem( xmlid );"
58     )
60     droptable :frazza_anot
62     przepisz_tabele( "create TABLE IF NOT EXISTS interpretacja(
           interpretacja_id integer primary key autoincrement, token_id
           integer not null, lex_xmlid text not null, msd_xmlid text not
           null, path_id integer not null, numer_lex_token integer,
           reszta_tagu text, leksem_id integer, disamb boolean default
           'f', dodana boolean default 'f', cz_m_leksem_id integer, sensy_id
           integer, created_at timestamp, updated_at timestamp );",
64         "create index IF NOT EXISTS
           interpretacja_leksem_idon interpretacja( leksem_id
           );",
66         "create index IF NOT EXISTS interpretacja_token_idon
           interpretacja( token_id );",
68         "create index if not exists interpretacja_disambon
           interpretacja( disamb );",
70         "create index if not exists
           interpretacja_token_id_disambon interpretacja(
           token_id, disamb );",
72         "create unique index IF NOT EXISTS
           interpretacja_path_id_msd_xmlidon interpretacja(
           path_id, msd_xmlid );"
73     )
75     przepisz_tabele( "create TABLE IF NOT EXISTS interpretacja_anot (
           interpretacja_anot_id integer primary key autoincrement,
           akapit_transzy_id integer not null, uzytkownik_id integer
           not null, token_id integer not null,
           interpretacja_id integer not null, created_at timestamp,
           updated_at timestamp);",
77         "create index IF NOT EXISTS
           interpretacja_anot_interpretacja_id
           on interpretacja_anot( interpretacja_id );",

```

```

79      "create index IF NOT EXISTS
      interpretacja_anot_token_id                        on
      interpretacja_anot (token_id);",

81      "create UNIQUE index if not exists
      interpretacja_anot_token_id_akapit_transzy_id
      on interpretacja_anot (token_id,
      akapit_transzy_id);",

83      "create index IF NOT EXISTS
      interpretacja_anot_uzytkownik_id
      on interpretacja_anot (uzytkownik_id);",

85      "create index IF NOT EXISTS
      interpretacja_anot_token_id_uzytkownik_id
      on interpretacja_anot (token_id, uzytkownik_id);"
86  )

89  przepisz_tabele( "create TABLE IF NOT EXISTS
      klasa_gram(klasa_gram_id integer primary key autoincrement,cz_m_id
      integer,klasa_gram_nazwa text,klasa_gram_ozn text,klasa_sem
      text,created_at timestamp,updated_at timestamp);",

91      "create unique index IF NOT EXISTS
      klasa_gram_klasa_gram_oznon klasa_gram
      (klasa_gram_ozn);"
92  )

95  przepisz_tabele( "create TABLE IF NOT EXISTS
      komentarz(komentarz_id integer primary key autoincrement,akapit_id
      integer,uzytkownik_id integer,akapit_transzy_id
      integer,przyczyna_odrzucenia integer default 0,tresc text,nowy
      boolean default 't',created_at timestamp,updated_at timestamp);",

97      "create index IF NOT EXISTS
      komentarz_uzytkownik_id_akapit_idon
      komentarz(uzytkownik_id, akapit_id);",

99      "create index IF NOT EXISTS
      komentarz_akapit_transzy_idon
      komentarz(akapit_transzy_id);",

101     "create index IF NOT EXISTS komentarz_akapit_idon
      komentarz(akapit_id);",

103     "create index IF NOT EXISTS komentarz_created_aton
      komentarz(created_at);"
104  )

107  przepisz_tabele( "create TABLE IF NOT EXISTS leksem(leksem_id
      integer primary key autoincrement,lex_xmlid text not null,path_id
      integer not null, - to ma sens: do utozsamienia z~eltem \\rhtml{<fs
      type="lex">} w~Korpusie.lemat text,klasa_gram_id integer,cz_m_id
      integer, - do sensowania (redundantne, 0+)cz_m_leksem_id integer,
      - do sensowania (0+)created_at timestamp,updated_at timestamp);",

```

```

109         "create unique index if not exists
        leksem_path_id_lex_xml_id on leksem( path_id,
        lex_xml_id );",

111         "create index IF NOT EXISTS
        leksem_lemat_klasa_gram_id on leksem (lemat,
        klasa_gram_id);",

113         "create unique index if not exists
        leksem_path_id_lemat_klasa_gram_id on leksem(
        path_id, lemat, klasa_gram_id );"
114     )

116     przepisz_tabele( "create TABLE IF NOT EXISTS
        morphosyntactic_rozbieznosc(          morphosyntactic_rozbieznosc_id
        integer primary key autoincrement,          token_id integer not null,
        interpretacja_id integer not null,          reszta_tagu text,
        leksem_id integer not null, created_at timestamp, updated_at
        timestamp);",

118         "create index IF NOT EXISTS
        morphosyntactic_rozbieznosc_token_id      on
        morphosyntactic_rozbieznosc( token_id );"
119     )

121     przepisz_tabele( "create TABLE IF NOT EXISTS prosby_anotatorek(
        prosby_anotatorek_id integer primary key autoincrement,
        dotyczy_id integer not null,          dotyczy_type text not null,
        uzytkownik_id integer not null,          opis text,          created_at
        timestamp,          updated_at timestamp,          rozpatrzona boolean
        default 'f',          spelniona boolean);",

123         "create unique index if not exists
        prosby_anotatorek_dotyczy_type_dotyczy_id on
        prosby_anotatorek( dotyczy_type, dotyczy_id);"
124     )

126     przepisz_tabele( "create table if not exists nowa_segmentacja(
        nowa_segmentacja_id integer primary key autoincrement,          ids
        text,          total_akapitow integer, - == (select count(*) from
        akapit) ensured by a trigger          seg_is text,          seg_should_be
        text,          nps boolean not null default 't',          wprowadzona
        boolean default 'f');",

128         "create unique index if not exists
        nowa_segmentacja_seg_is_seg_should_be_ids_npson
        nowa_segmentacja( seg_is, seg_should_be, ids, nps
        );"
129     )

131     ##      execute "drop trigger  nowa_segmentacja_total_akapitow"
133     # trigger został unicestwiony wraz z tabelą
134     execute "create trigger  nowa_segmentacja_total_akapitow
        after insert on nowa_segmentacja begin  update nowa_segmentacja set

```

```

total_akapitow = (select count(*) from akapit) where
nowa_segmentacja_id = new.nowa_segmentacja_id;end;"

137  przepis_z_tabele( "create TABLE IF NOT EXISTS
      poziomy_anotacji(poziomy_anotacji_id integer primary key
      autoincrement,uzytkownik_id integer not null,segmentation boolean
      default 'f',sentences boolean default 'f',morphosyntactic boolean
      default 'f',word_senses boolean default 'f',syntactic_words boolean
      default 'f',named_entities boolean default 'f',syntactic boolean
      default 'f',created_at timestamp,updated_at timestamp);"

139  "create unique index if not exists poziomy_anotacji_uzytkownik_idon
      poziomy_anotacji( uzytkownik_id );"
140  )

142  przepis_z_tabele( "create TABLE IF NOT EXISTS protokol(
      protokol_id integer primary key autoincrement,          poziom text not
      null,          akapit_id integer not null,          czy_rozbieznosc
      boolean not null default 't',czy_superancja boolean not null default
      'f',          created_at timestamp,updated_at timestamp);"

144  "create index IF NOT EXISTS protokol_created_aton protokol(
      created_at );",

146  "create index IF NOT EXISTS protokol_czy_rozbieznosc          on protokol(
      czy_rozbieznosc );",

148  "create index IF NOT EXISTS protokol_poziom          on protokol( poziom
      );",

150  "create index IF NOT EXISTS protokol_akapit_id          on protokol(
      akapit_id );"
151  )

154  przepis_z_tabele( "create TABLE IF NOT EXISTS punkt_protokolu(
      punkt_protokolu_id integer primary key autoincrement,
      elt_protokol_id integer not null,          elt_protokol_type text not
      null,          protokol_id integer not null,          akapit_transzy_id
      integer not null,          uzytkownik_id integer,created_at
      timestamp,updated_at timestamp);"

156  "create index IF NOT EXISTS
      punkt_protokolu_elt_protokol_id_elt_protokol_type          on
      punkt_protokolu( elt_protokol_id, elt_protokol_type );",

158  "create index IF NOT EXISTS punkt_protokolu_protokol_id          on
      punkt_protokolu( protokol_id );",

160  "create index IF NOT EXISTS punkt_protokolu_uzytkownik_id          on
      punkt_protokolu( uzytkownik_id );",

162  "create index IF NOT EXISTS punkt_protokolu_akapit_transzy_id          on
      punkt_protokolu( akapit_transzy_id );"
163  )

```

```
166  przepisz_tabelę( "create TABLE IF NOT EXISTS rola(      opis text,
      rola_id integer primary key, opis_kr string, created_at timestamp,
      updated_at timestamp);" )

169  przepisz_tabelę( "create table if not exists sg_choice(
      sg_choice_id integer primary key autoincrement,      akapit_id
      integer not null, - redundantne, dla łatwiejszego wyszukiwania
      dodany boolean default 'f',      xmlid text,      created_at
      timestamp,      updated_at timestamp      );",

171  "create index if not exists sg_choice_akapit_idon sg_choice(
      akapit_id );"

172  )

175  przepisz_tabelę( "create table if not exists sg_variant_annot(
      sg_variant_annot_id integer primary key autoincrement,
      sg_choice_id integer not null,      sg_variant_id integer not null,
      chosen boolean not null,      akapit_transzy_id integer not null,
      uzytkownik_id integer,      created_at timestamp,      updated_at
      timestamp);" ,

177  "create unique index if not exists - żeby wymusić jednoznaczność
      ujednoznacznienia segmentacji
      sg_variant_annot_akapit_transzy_id_sg_variant_idon sg_variant_annot(
      akapit_transzy_id, sg_variant_id);"

178  )

181  przepisz_tabelę( "create table if not exists sg_variant(
      sg_variant_id integer primary key autoincrement,      sg_choice_id
      integer not null,      xmlid text,      dodany boolean default 'f',
      created_at timestamp,      updated_at timestamp);" ,

183  "create index if not exists sg_variant_sg_choice_idon sg_variant(
      sg_choice_id );"

184  )

186  przepisz_tabelę( "create TABLE IF NOT EXISTS segmentation_rozbieznosc(
      segmentation_rozbieznosc_id integer primary key autoincrement,
      sg_choice_id integer not null,      sg_variant_id integer not
      null,created_at timestamp,updated_at timestamp);" ,

188  "create index IF NOT EXISTS segmentation_rozbieznosc_sg_choice_id
      on segmentation_rozbieznosc( sg_choice_id );"

189  )

192  przepisz_tabelę( "create table if not exists koniec_zdania_annot(
      koniec_zdania_annot_id integer primary key autoincrement,
      akapit_id integer not null,      akapit_transzy_id integer not null,
      uzytkownik_id integer not null,      token_id integer not null,
      created_at timestamp,      updated_at timestamp      );",

194  " create unique index if not exists
      koniec_zdania_annot_akapit_transzy_id_token_idon koniec_zdania_annot(
      akapit_transzy_id, token_id );",
```

```
196 " create index if not exists koniec_zdania_anot_akapit_idon
    koniec_zdania_anot( akapit_id ); "
197 )

200 przepisz_tabele( "create table if not exists sentences_rozbieznosc(
    sentences_rozbieznosc_id integer primary key autoincrement,
    token_id integer not null,      czy_konczy_zdanie boolean not null,
    created_at timestamp,      updated_at timestamp);" ,

202 "create index if not exists sentences_rozbieznosc_token_idon
    sentences_rozbieznosc( token_id );"
203 )

205 przepisz_tabele(
206 "create TABLE IF NOT EXISTS statusy(      statusy_id integer primary
    key autoincrement,      akapit_transzy_id integer not null,
    akapit_id integer not null,      segmentation integer default -1,
    sentences integer default -1,      morphosyntactic integer default -1,
    word_senses integer default -1,      syntactic_words integer default
    -1,      named_entities integer default -1,      syntactic integer
    default -1,      segmentation_uzid integer,      sentences_uzid
    integer,      morphosyntactic_uzid integer,      word_senses_uzid
    integer,      syntactic_words_uzid integer,      named_entities_uzid
    integer,      syntactic_uzid integer,      segmentation_odrzucony
    boolean default 'f',      sentences_odrzucony boolean default 'f',
    morphosyntactic_odrzucony boolean default 'f',
    word_senses_odrzucony boolean default 'f',
    syntactic_words_odrzucony boolean default 'f',
    named_entities_odrzucony boolean default 'f',
    syntactic_odrzucony boolean default 'f',      created_at timestamp,
    updated_at timestamp);" ,

208 "create index IF NOT EXISTS statusy_morphosyntactic on
    statusy(morphosyntactic);" ,
209 "create index IF NOT EXISTS statusy_named_entities on
    statusy(named_entities);" ,
210 "create index IF NOT EXISTS statusy_segmentation on
    statusy(segmentation);" ,
211 "create index IF NOT EXISTS statusy_sentences on
    statusy(sentences);" ,
212 "create index IF NOT EXISTS statusy_syntactic on
    statusy(syntactic);" ,
213 "create index IF NOT EXISTS statusy_syntactic_words on
    statusy(syntactic_words);" ,
214 "create index IF NOT EXISTS statusy_word_senses on
    statusy(word_senses);" ,

216 "create UNIQUE index if not exists statusy_akapit_transzy_idon
    statusy( akapit_transzy_id );"
217 )
```



```

219  przepisz_tabele( "create TABLE IF NOT EXISTS tagset(      tagset_id
      integer primary key autoincrement,      typ text,      lewe text,
      prawe text,      created_at timestamp,      updated_at timestamp);",
221  "create index IF NOT EXISTS tagset_typ_leweon tagset(typ, lewe);"
222  )

225  przepisz_tabele( "create TABLE IF NOT EXISTS token(  token_id integer
      primary key autoincrement,  kolejnosc integer, - not null ensured by
      a trigger  akapit_id integer not null,  xpointer text,
      segmentation_xmlid text,  morphosyntactic_xmlid text,
      fs_morph_comment text,  path_id integer not null,  orth text,
      czy_interp boolean not null,  ns_poprzedza boolean default 'f',
      ns_nastepuje boolean,  czy_konczy_zdanie boolean,
      czy_konczy_zdanie_updated_at timestamp,  sg_choice_id integer null
      default null,  sg_variant_id integer null default null,  dodany
      boolean default 'f' not null,  chosen boolean default 't', -
      dedukowalne z sg_variant_anot  chosen_updated_at timestamp,
      superancja text, - zserializowana tablica #\{[\<poz_sym>*\]\}
      created_at timestamp,  updated_at timestamp);",

227  "create unique index if not exists token_kolejnoscon token( kolejnosc
      );",

229  "create index if not exists token_ns_poprzedzaon token( ns_poprzedza
      );",

231  "create index if not exists token_ns_nastepujeon token( ns_nastepuje
      );",

233  "create index IF NOT EXISTS token_akapit_idon  token( akapit_id );",
235  "create unique index if not exists token_path_id_segmentation_xmlidon
      token( path_id, segmentation_xmlid );",

237  "create index if not exists token_path_id_morphosyntactic_xmlidon
      token( path_id, morphosyntactic_xmlid );",

239  "create index if not exists token_sg_choice_idon token( sg_choice_id
      );",

241  "create index if not exists token_sg_variant_idon token(
      sg_variant_id );",

243  "create index if not exists token_czy_interp on token( czy_interp );",
245  "create index - nie unique, bo tylko do wyszukiwania (przy
      danowlewie)if not exists token_akapit_id_segmentation_xmlid_orthon
      token( akapit_id, segmentation_xmlid, orth );"
246  )

248  ## execute "drop trigger token_default_kolejnosc"
250  # trigger unicestwiony wraz z tabelą
251  execute "create trigger  token_default_kolejnosc      after insert
      on tokenwhen (select kolejnosc from token where token_id =
      new.token_id) is nullbeginupdate token set kolejnosc = (new.token_id
      * 216)where token_id = new.token_id;end;"

```

```

254  przepis_tabele( "create TABLE IF NOT EXISTS transza(  transza_id
integer primary key autoincrement,  uzytkownik_id integer,
created_at timestamp,  updated_at timestamp);",

256  "create index IF NOT EXISTS transza_uzytkownik_id          on transza
(uzytkownik_id);"
257  )

259  przepis_tabele(
260  "create TABLE IF NOT EXISTS akapit_transzy(  akapit_transzy_id
integer primary key autoincrement,  akapit_id integer not null,
transza_id integer not null,  status integer default 0 not null,
odrzucony boolean default 'f',  uzytkownik_id integer,  created_at
timestamp,  updated_at timestamp,  blizniaczy_id integer);",

262  "create index IF NOT EXISTS akapit_transzy_status          on
akapit_transzy( status );",

264  "create index IF NOT EXISTS akapit_transzy_transza_id      on
akapit_transzy( transza_id );",

266  "create index IF NOT EXISTS akapit_transzy_uzytkownik_id   on
akapit_transzy( uzytkownik_id );",

268  "create index IF NOT EXISTS akapit_transzy_akapit_id       on
akapit_transzy( akapit_id );",

270  "create index IF NOT EXISTS akapit_transzy_odrzucony       on
akapit_transzy( odrzucony );"
271  )

273  przepis_tabele(
274  "create TABLE IF NOT EXISTS akapit(  akapit_id integer primary key
autoincrement,  morph_id integer not null,  path_id integer not
null, - używane przy wyszukiwaniu akapitu podczas wczytywania
\\file{sg.xml}.  segmentation_xmlid text,  segmentation_xlink_href
text,  morphosyntactic_xmlid text,  created_at timestamp,
updated_at timestamp  );",

276  "create index if not exists akapit_morph_idon akapit( morph_id );",
278  "create index if not exists akapit_path_idon akapit( path_id );",
280  "create unique index if not exists akapit_path_id_segmentation_xmlidon
akapit( path_id, segmentation_xmlid );",

282  "create unique index if not exists
akapit_path_id_morphosyntactic_xmlidon akapit( path_id,
morphosyntactic_xmlid );"
283  )

285  przepis_tabele(
286  "create table if not exists morph(          morph_id integer primary key
autoincrement,          xmlid text not null,          path_text text not
null, - identical as in table \\sql{path}          created_at timestamp,
updated_at timestamp);",

```

```

288   "create unique index if not exists morph_xmlid_path_text on morph(
      xmlid, path_text );"
289   )

291   przepisz_tabele(
292     "create table if not exists path(          path_id integer primary key
      autoincrement,          path_text text not null,          created_at
      timestamp,          updated_at timestamp          );",
294     "create unique index if not exists path_path_text on path( path_text
      );"
295   )

298   przepisz_tabele( "CREATE TABLE uzy.uzytkownik (          uzytkownik_id
      integer primary key autoincrement,          rola_id integer not null,
      login text not null,          hasz_haslo text not null,          sol text
      not null,          imie text,          nazwisko text,          email text,
      jak_annotator boolean default 'f', - dla zarządcy: czy ma mieć
      dostępność poziomów jak annotator, tj. po kolei w miarę
      zatwierdzania, czy wszystkich naraz.          created_at timestamp ,
      updated_at timestamp,          haslo_updated_at timestamp);",

300           [ :login, :unique ]
301   )

303   pragmas_normal
304     Uzytkownicy8003.connection.execute "vacuum"
305     execute "vacuum"
306     end # of if false

308   end # of self.up

310   def self.down
311     end
312   end

```

### 019\_popraw\_tok\_xpointers.rb

```

23   class PoprawTokXpointers < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28
29       addcolumn :nowa_segmentacja, :akapit_transzy_id, :integer
30       addindex :nowa_segmentacja, :akapit_transzy_id
31
32       addcolumn :nowa_segmentacja, :transza_id, :integer
33       addindex :nowa_segmentacja, :transza_id
34
35       tokproc = Proc.new {|t|
36         puts "### t. #{t.id} »#{t.orth}« #{t.xpointer}"
37
38         # sprawdzimy, czy nasz sg_choice ma jakiegokolwiek tokeny wybrane
39         i ew. poprawimy
40         sgc = t.sg_choice
41         sgct= sgc.token

```

```

42     jest_wybrany = false
43     sgct.each { |t0| if t0.chosen? : jest_wybrany = true end }
44
45     unless jest_wybrany
46       # jeśli wszystkie warianty oznaczyło jako odrzucone, to wybieramy
       # ten, który ma najstarsze tokeny, co z dobrym przybliżeniem
       # odpowiada minimalnemu idowi.
47       puts "#{t.id} »#{t.orth}«ma sg_choice cały odrzucony!
       Poprawiam"
48
49       Token.find( sgct.tokids.min ).sg_variant.token.each { |t1|
50         t1.chosen = true
51         t1.save!
52       }
53     end
54
55     tp = t.poprzedni
56     md = /^(.* ),(\d+),(\d+)\)$/ .match( tp.xpointer )
57     if md
58       tp_od_znaku = md[ 2 ].to_i # $2
59       tp_ile_znaków = md[ 3 ].to_i # $3
60       t_od_znaku = tp_od_znaku + tp_ile_znaków + if t.ns_poprzedza? :
       0 else 1 end
61
62       t_ile_znaków = t.orth.jlength
63     end
64
65     t.xpointer = md[ 1 ].gsub( '-', '- ' )
66     + ",#{t_od_znaku},#{t_ile_znaków}"
67     t.save!
68     puts "t. #{t.id} fixed to #{Token.find( t.id ).xpointer}"
69   } # of tokproc
70
71   if AnoVersion.port == 8003 and t169 = Token.
72     find( :first, :conditions => { :token_id => 169274 } )
73     tokproc.call( t169 )
74   end
75
76   Token.find( :all, :conditions => { :dodany => true }, :order =>
77     :kolejnosc ).each { |tok|
78     tokproc.call( tok )
79   }
80
81   if AnoVersion.port == 8003
82     ProsbyAnotatorek.find( :all, :conditions => {
83       :prosby_annotatorek_id => [36, 100] } ).each { |pro|
84       if pro.updated_at < '2009-07-28 0:00'.to_time
85         pro.rozpatrzona = false
86         pro.save!
87       end
88     }
89   end # of if 8003
90   end # of self.up
91
92   def self.down

```

```
91     end
92 end
```

### **o2o\_popraw\_komentarze.rb**

```
23 class PoprawKomentarze < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     pragmas_for_update
29     execute "update komentarz set akapit_id=(select akapit_id from
        akapit_transzy where
        akapit_transzy_id=komentarz.akapit_transzy_id) where akapit_id is
        null"
30     pragmas_normal
31   end
32
33   def self.down
34     end
35 end
```

### **o21\_add\_uzytkownik\_tylko\_nowe\_komentarze.rb**

```
23 class AddUzytkownikTylkoNoweKomentarze < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     pragmas_for_update
29
30     addcolumn :uzytkownik, :tylko_nowe_komentarze_oglactwo, "boolean
        default 't'"
31     addcolumn :uzytkownik, :tylko_nowe_komentarze_anotacja, "boolean
        default 'f'"
32     pragmas_normal
33   end
34
35   def self.down
36     end
37 end
```

### **o22\_add\_dziubnij\_w\_segmentacjach.rb**

```
23 class AddDziubnijWSegmentacjach < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     pragmas_for_update
29
30     if Time.now < '2009-07-29 03:37'.to_time
31       idy = [1053, 1339, 1340, 1606, 1684]
32
33       NowaSegmentacja.find_all.each { |ns|
34         if (ns.ids & idy)[0]
35           puts "### ns #{ns.id}"
36         end
37       }
38     end
39   end
40 end
```

```

36         puts ns.seg_is.inspect + " -> " + ns.seg_should_be.inspect
37         ns.wprowadzona = false
38         ncomm = ns.commit
39         puts " wystąpiłam w ak. #{(ncomm && ncomm[:akids]).inspect}
40         \n"
41     end
42 }

44 atytr = AkapitTranszy.find( :all, :conditions => {:akapit_id =>
45     idy } )
46 atytr.each {|akat|
47     puts "### akapit #{akat.akapit_id}: atr #{akat.id}, tr
48     #{akat.transza_id}"
49     akat.set_status( :segmentation, :dopuszczony, :tylko_ten )
50 }
51 Akapit.find( idy ).each { |ak|
52     ak.token_rejected.each { |t|
53         t.chosen = true
54         t.save!
55     }
56 }

57 end
58 pragmas_normal

60 def self.down
61 end
62 end

```

### o23\_popraw\_xpointer\_przecinek.rb

```

23 class PoprawXpointerPrzecinek < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28         pragmas_for_update
29         Token.find( :all, :conditions => { :dodany => true } ).each {
30             |t182|
31             if (xp182 = t182.xpointer) =~ /^(.* ab)(\d+),(\d+)\)/
32                 t182.xpointer = "#{$1},#{ $2},#{ $3}"
33                 t182.save!
34                 puts xp182 + ' -> ' + t182.xpointer
35             end
36         }
37         pragmas_normal
38     end
39
40     def self.down
41     end
42 end

```

**024\_dziubnij\_w\_segmentacjach.rb**

```

23  class DziubnijWSegmentacjach < ActiveRecord::Migration
24
25      extend MigrationHelper
26
27      def self.up
28          pragmas_for_update
29
30          if Time.now < '2009-07-31 23:37'.to_time
31              idys = [ 1545 ]
32              idy = [ 1513, 1545 ]
33              NowaSegmentacja.find_all.each { |ns|
34                  if (ns.ids & idys)[0]
35                      puts "### ns #{ns.id}"
36                      puts ns.seg_is.inspect + " -> " + ns.seg_should_be.inspect
37                      ns.wprowadzona = false
38                      ncomm = ns.commit
39                      puts " wystąpiłam w ak. #{(ncomm && ncomm[:akids]).inspect}
40                          \n"
41                  end
42              }
43
44              atytr = AkapitTranszy.find( :all, :conditions => {:akapit_id =>
45                  idy } )
46              atytr.each { |akat|
47                  puts "### akapit #{akat.akapit_id}: atr #{akat.id}, tr
48                      #{akat.transza_id}"
49                  akat.set_status( :segmentation, :dopuszczony, :tylko_ten )
50              }
51              Akapit.find( idy ).each { |ak|
52                  ak.token_rejected.each { |t|
53                      t.chosen = true
54                      t.save!
55                  }
56              }
57          end
58          pragmas_normal
59      end
60
61      def self.down
62          end
63      end

```

**025\_dziubnij\_w\_segmentacjach.rb**

```

23  class DziubnijWSegmentacjach < ActiveRecord::Migration
24
25      extend MigrationHelper
26
27      def self.up

```

```

28     pragmas_for_update

30     if Time.now < '2009-07-31 23:37'.to_time
31         idys = [ 1734 ]
32         idy = [ 1483, 1531, 1734 ]

34         idys.each { |id183|
35             NowaSegmentacja.find_all.each { |ns|
36                 if ns.ids == [ id183 ]
37                     puts " akapit #{id183}, ns #{ns.id}"
38                     puts " @@@ ns #{ns.id}"
39                     puts ns.seg_is.inspect + " -> " + ns.seg_should_be.inspect
40                     ns.wprowadzona = false
41                     ncomm = ns.commit
42                     puts " wystąpiłam w ak. #{(ncomm && ncomm[:akids]).inspect}
                       \n"
43                 end
44             }
45         }

47         atytr = AkapitTranszy.find( :all, :conditions => { :akapit_id =>
            idy } )
48         atytr.each { |akat|
49             puts " @@@ akapit #{akat.akapit_id}: atr #{akat.id}, tr
              #{akat.transza_id}"
50             akat.set_status( :segmentation, :dopuszczony, :tylko_ten )
51         }
52         Akapit.find( idy ).each { |ak|
53             ak.token_rejected.each { |t|
54                 t.chosen = true
55                 t.save!
56             }
57         }

59     end
60     pragmas_normal
61 end

63 def self.down
64     end
65 end

```

## 026\_zasegmentuj\_dolewke.rb

```

23 class ZasegmentujDolewke < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28         kill_anotatorni_if_8003
29
30         Akapit.autozatwierdź_segmentation
31
32     end

```



```
34     def self.down
35     end
36 end
```

### 027\_popraw2968.rb

```
23 class Popraw2968 < ActiveRecord::Migration
24   extend MigrationHelper

26   def self.up
27     pragmas_for_update

29     akiki = Akapit.find_by_sql "select * from akapit where (select
      count(*) from token t where t.akapit_id=akapit.akapit_id)=0"

31     akiki.each { |a184|
32       PoziomyAnotacji.poziomy_x.each { |poz184|
33         a184.akat0.set_status( poz184, :osadzony, false,
          Uzytkownik.natror.id )
34       } # of each poziom
35     } # of each empty akapit
36     pragmas_normal
37   end

39   def self.down
40   end
41 end
```

### 028\_cz\_m\_leksem\_id\_indices.rb

```
23 class CzMLeksemIdIndices < ActiveRecord::Migration
24   extend MigrationHelper

26   def self.up
27     pragmas_for_update

29     addindex :interpretacja, :cz_m_leksem_id
30     addindex :leksem, :cz_m_leksem_id

32     # taka parka indeksów to przy 4118 akapitach ok. 38 MB (na 412/450
      całej bazy).
33     ##      addindex :interpretacja, :updated_at
34     ##      addindex :leksem, :updated_at

38     addindex :interpretacja, :created_at
39     addindex :leksem, :created_at

42     pragmas_normal
43   end

45   def self.down
46   end
47 end
```

**030\_sens\_anot\_dopisz\_token\_id.rb**

```

23  class SensAnotDopiszTokenId < ActiveRecord::Migration
24
25      extend MigrationHelper
26
27      def self.up
28          pragmas_for_update
29
30          unless SensAnot.column_names.include?( "token_id" )
31              if SensAnot.count == 0
32                  przepis_tabele(
33
34                      'CREATE TABLE IF NOT EXISTS sens_anot(- Attributes
35                      -   sens_anot_id integer primary key autoincrement,
36                        uzytkownik_id integer,   akapit_transzy_id integer
37                        not null,   token_id integer not null,
38                        interpretacja_id integer not null,   sensy_id
39                        integer not null,   automatycznie boolean,
40                        created_at timestamp, - default current_timestamp,
41                        updated_at timestamp);',
42
43                      'CREATE INDEX IF NOT EXISTS sens_anot_sensy_idon sens_anot( sensy_id
44                      );',
45
46                      'CREATE INDEX IF NOT EXISTS sens_anot_interpretacja_idon sens_anot(
47                      interpretacja_id );',
48
49                      'CREATE INDEX IF NOT EXISTS sens_anot_token_idon sens_anot( token_id
50                      );',
51
52                      'CREATE INDEX IF NOT EXISTS
53                      sens_anot_interpretacja_id_akapit_transzy_idon sens_anot(
54                      interpretacja_id, akapit_transzy_id );',
55
56                      'CREATE INDEX IF NOT EXISTS sens_anot_akapit_transzy_idon sens_anot(
57                      akapit_transzy_id );',
58                      )
59                      errmess = "SensAnot niepusty! nie mogę przepisać!"
60                      puts errmess
61                      else raise errmess
62                      end
63                  end
64
65                  addindex :sens_anot, [ :akapit_transzy_id, :token_id] , :unique
66
67                  pragmas_normal
68              end
69
70      def self.down
71          end
72      end

```

**031\_add\_czmleksem\_dopisany.rb**

```
23 class AddCzmleksemDopisany < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     kill_anotatorni _if_8003
29
30     addcolumn :leksem, :cz_m_przypisana, "boolean default 'f' "
31     addindex :leksem, :cz_m_przypisana
32
33     addcolumn :interpretacja, :cz_m_leksem_przypisany, "boolean
34       default 'f' "
35     addindex :interpretacja, :cz_m_leksem_przypisany
36
37     addcolumn :leksem, :cz_m_leksem_przypisany, "boolean default 'f' "
38     addindex :leksem, :cz_m_leksem_przypisany
39
40   end
41
42   def self.down
43   end
44 end
```

**032\_create\_wsd\_rozbieznosc.rb**

```
23 class CreateWsdRozbieznosc < ActiveRecord::Migration
24
25   def self.up
26     execute "CREATE TABLE IF NOT EXISTS word_senses_rozbieznosc(
27       word_senses_rozbieznosc_id integer primary key
28       autoincrement,token_id integer not null,          sensy_id integer
29       not null,created_at timestamp,updated_at timestamp);"
30
31     execute "CREATE INDEX IF NOT EXISTS
32       word_senses_rozbieznosc_token_id      on word_senses_rozbieznosc(
33       token_id );"
34
35   end
36
37   def self.down
38   end
39 end
```

**033\_sensy\_add\_opishtml.rb**

```
23 class SensyAddOpishtml < ActiveRecord::Migration
24
25   extend MigrationHelper
26
27   def self.up
28     # a no-op wskutek cofnięcia do v.10 i remigracji wszystkiego
29     z poprawionym utworzeniem tabeli sensy.
30     addcolumn :sensy, :long_def_html, :text
31     addcolumn :sensy, :n, :integer
32
33   end
34 end
```

```
31     end
33     def self.down
34     end
35 end
```

#### **034\_statusy\_index\_akapitid.rb**

```
23 class StatusyIndexAkapitid < ActiveRecord::Migration
25   extend MigrationHelper
27   def self.up
28     addindex :statusy, :akapit_id
29   end
31   def self.down
32   end
33 end
```

#### **035\_akapit\_add\_tresc.rb**

```
23 class AkapitAddTresc < ActiveRecord::Migration
25   extend MigrationHelper
27   def self.up
28     pragmas_for_update
29     addcolumn :akapit, :tresc, :text
30     addcolumn :akapit, :incipit, :text
32     Akapit.reset_column_information
34     Akapit.find_all.each { |aka195|
35       aka195.update_treści
36     }
38     pragmas_normal
39   end
41   def self.down
42   end
43 end
```

#### **036\_statusy\_add\_transzaid.rb**

```
23 class StatusyAddTranszaid < ActiveRecord::Migration
25   extend MigrationHelper
27   def self.up
28     pragmas_for_update
29     addcolumn :statusy, :transza_id, :integer
30     addindex :statusy, :transza_id
31     execute "update statusy set transza_id=(select transza_id from
      akapit_transzy at where
      at.akapit_transzy_id=statusy.akapit_transzy_id)"

```

```

32     pragmas_normal
33   end
34
35   def self.down
36     end
37   end

```

### **o37\_statusy\_popraw\_wsd.rb**

```

23   class StatusyPoprawWsd < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28       ak_od_do = Token.find_by_sql( " select min( akapit_id) as ak_od,
                                     max( akapit_id ) as ak_do from token where exists (select
                                     interpretacja_id from interpretacja i where cz_m_leksem_id is not
                                     null and i.token_id=token.token_id and disamb='t') and (not exists
                                     (select sens_anot_id from sens_anot sa where
                                     sa.token_id=token.token_id)) and (select min(word_senses) from
                                     statusy s where s.akapit_id=token.akapit_id)>=16 ;" ).collect {
                                     |t198| [t198.ak_od.to_i, t198.ak_do.to_i ] }.flatten
29
30       kill_anotatorni _if_8003
31
32       execute "update statusy set word_senses=0 where akapit_id in
               (select distinct akapit_id from token where exists (select
               interpretacja_id from interpretacja i where cz_m_leksem_id is not
               null and i.token_id=token.token_id and disamb='t') and (not exists
               (select sens_anot_id from sens_anot sa where
               sa.token_id=token.token_id)) and (select min(word_senses) from
               statusy s where s.akapit_id=token.akapit_id)>=16) ;"
33
34       AkapitTranszy.update_all_Stati( * ak_od_do )
35
36     end
37
38     def self.down
39       end
40   end

```

### **o38\_uspojnij\_path\_id.rb**

```

23   class UspojnijPathId < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28       kill_anotatorni _if_8003
29       if Token.find_by_sql( "select count(*) as ile from token inner
                             join akapit using( akapit_id ) where
                             token.path_id<>akapit.path_id" )[0].ile.to_i != 0
30
31         puts "038 niespójność ścieżek!"
32
33         execute "update token set path_id=(select path_id from akapit
               where akapit_id=token.akapit_id) where path_id <> (select
               path_id from akapit where akapit_id=token.akapit_id)"

```

```

34     end
35     # powyższe szczęśliwie jest niepotrzebne: w tokenach się zgadza.

37     # teraz poprawimy tokeny zweryfikowane bez dezambiguacji

39     Token.find_by_sql( "select token.*, akapit_id from statusy inner
join token using(akapit_id) where chosen='t' and
morphosyntactic>=16 and not exists ( select * from interpretacja
where token_id=token.token_id and disamb='t') " ).each { |t200|
40         # wersja SQLite'a czy też adaptera na Chopinie nie obsługuje
nawiasowania joinów (nawiasowanie zakrywa nazwę tabeli)
41         if ta200 = t200.akapit
42             ta200.akat0.set_status( :morphosyntactic, :do_osądzania )
43         else
44             puts "#{t200.id}: #{t200.orth} bez akapittu!"
45             sleep 1
46         end
47     }

49     if Interpretacja.find_by_sql( "select count(*) as ile from
interpretacja inner join token using( token_id ) where
token.path_id<>interpretacja.path_id" )[0].ile.to_i != 0

51         InterpretacjaAnot.
52         find_by_sql(
53             "select interpretacja_anot.*, akapit_transzy_id,
interpretacja_id, interpretacja_anot.token_id " +
54             " from " +
55             " statusy inner join interpretacja_anot using(
akapit_transzy_id) " +
56             " inner join interpretacja using( interpretacja_id )
" +
57             " inner join token using( token_id ) " +
58             " where morphosyntactic>=16 and chosen='t' and
disamb='f' " ).

59         # jeśli jest interpretacja anot wskazująca w zatwierdzonym akapicie
nie-dezambiguację, to ją przewskazemy:
60         each { |ia200|
61             t200 = ia200.token
62             puts t200.id
63             ia200.interpretacja_id = t200.the_disambiguation.id
64             ia200.save!
65         }

67         execute "delete from interpretacja where dodana='t' and
disamb='f' and " +
68             " (not exists (select interpretacja_id from interpretacja_anot
" +
69             " where interpretacja_id=interpretacja.interpretacja_id )) and
" +
70             " (select count(*) from interpretacja i1 where
leksem_id=interpretacja.leksem_id " +

```

```
71      " and reszta_tagu=interpretacja.reszta_tagu and " +  
72      " token_id=interpretacja.token_id )>1 "  
  
74      execute "update interpretacja set path_id=(select path_id from  
token where token_id=interpretacja.token_id) where  
path_id<>(select path_id from token where  
token_id=interpretacja.token_id)"  
75 end # of niezgodność ścieżki w tokenie ze ścieżką w interpretacji  
  
77 # poprawimy przypisanie „leksemów” do interpretacyj, czyli niezg.  
ścieżki w leksemie ze ścieżką w interpretacji  
  
79 Interpretacja.  
80   find_by_sql(  
81     "select interpretacja.*, leksem_id from leksem inner  
join interpretacja using( leksem_id ) where  
interpretacja.path_id<>leksem.path_id"  
82   ).  
83 each { |int200|  
84   # schemat tej pętli pochodzi z metody Interpretacja.get_xmlids.  
85   path_id = int200.path_id  
86   le200 = int200.leksem  
  
88   puts "interpretacja #{int200.id}"  
  
90   unless le200  
91     puts "int. #{int200.id} bez leksemu!  
(leksem_id=#{int200.leksem_id.inspect})"  
92     sleep 5  
  
94   else  
  
96     if le201 = Leksem.find( :first,  
97                           :conditions =>  
98                             { :lemat => le200.lemat,  
99                               :klasa_gram_id => le200.klasa_gram_id,  
100                              :path_id => path_id })  
  
102       puts "          leksem jest"  
103       int200.leksem_id = le201.id  
104       int200.save!  
  
106     else # nie ma takiego leksemu w tej ścieżce – to bierzemy z innej  
ścieżki i duplikujemy, przypisując odpowiedni xmlid. Jakiś jest na  
pewno: wszak został już kiedyś przypisany.  
107       puts "          nowy leksem"  
108       t200 = int200.token  
  
lex_xmlid201 = Interpretacja.get_xmlids(  
110                                     t200.id,  
111                                     le200.lemat,  
112                                     le200.klasa_gram_id )[ 1 ]  
113  
115       le201 = Leksem.create!(  
116         ##           :leksem_id => le200.leksem_id
```

```

118             :lex_xmlid => lex_xmlid201,
119             :path_id => path_id,
120             :lemat => le200.lemat,
121             :klasa_gram_id => le200.klasa_gram_id,
122             :cz_m_id => le200.cz_m_id,
123             :cz_m_leksem_id => le200.cz_m_leksem_id
124         )
125         int200.leksem_id = le201.id
126         int200.save!
127     end
128     end # of ma leksem albo nie.
129 } # of each trefe interpretacja
131 if Leksem.find_by_sql( "select count(*) as ile from leksem inner
    join interpretacja using( leksem_id ) where
    interpretacja.path_id<>leksem.path_id " )[0].ile.to_i != 0
132     raise "A jednak nadal się nie zgadza!"
133 end
135 end # of up
137 def self.down
138     end
139 end

```

### **o39\_popraw\_dwa\_dywizy.rb**

```

23 class PoprawDwaDywizy < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28         pragmas_for_update
29         Token.find( :all, :conditions => "fs_morph_comment like '%-%' " ).
30             each { |t202|
31                 puts "#{t202.id}: #{t202.orth}"
32                 t202.fs_morph_comment = t202.fs_morph_comment.gsub( '-', '\-' )
33                 t202.save!
34             }
35
36     end
37
38     def self.down
39         end
40 end

```

### **o40\_uzupelnij\_dolewke.rb**

```

23 class UzupelnijDolewke < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28         kill_anotatorni_if_8003
29         Akapit.update_treści_null

```



```

30     Statusy uzupełnij_transza_id
31   end

33   def self.down
34     end
35 end

```

#### 041\_revert\_autoword.rb

```

23   class RevertAutoword < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28       # 2009/10/12 odkryłem potworną katastrofę, polegającą na tym, że po
        ms-zatwierdzeniu akapitu wsd zawsze zatwierdzały się same (bez anotacji).
        Dotknęło to prawie 2000 akatów.
29       kill_anotatorni _if_8003_nobackup
30
31       puts "\n" + Statusy.find_by_sql( "select count(*) as ile from
        statusy where word_senses=16 and exists (select interpretacja_id
        from interpretacja i inner join token tok using( token_id ) where
        i.cz_m_leksem_id is not null and i.disamb='t' and
        tok.akapit_id=statusy.akapit_id ) and not exists (select * from
        sens_anot where akapit_transzy_id=statusy.akapit_transzy_id);"
        )[0].ile + " trefnych.\n"
32
33       execute "update statusy set word_senses=0, word_senses_uzid=null,
        updated_at=current_timestamp where word_senses=16 and exists
        (select interpretacja_id from interpretacja i inner join token tok
        using( token_id ) where i.cz_m_leksem_id is not null and
        i.disamb='t' and tok.akapit_id=statusy.akapit_id ) and not exists
        (select * from sens_anot where
        akapit_transzy_id=statusy.akapit_transzy_id);"
34
35       AkapitTranszy.find_by_sql(
36         "select at.* from akapit_transzy at inner
        join statusy s using(akapit_transzy_id)
        where status>=16 and word_senses=0" ).
37       each { |akat205| akat205.update_Status }
38
39     end
40
41     def self.down
42     end
43 end

```

#### 042\_popraw\_nierowne\_statusy.rb

```

23   class PoprawNierowneStatusy < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28       kill_anotatorni _if_8003
29       sth_done = false

```

```

30     AkapitTranszy.find_by_sql(
31         "select * from akapit_transzy where
32         akapit_transzy_id in " +
33         "(select s1.akapit_transzy_id from " +
34         " statusy s1 inner join statusy s2 on " +
35         " s1.akapit_id=s2.akapit_id and
36         s1.statusy_id<>s2.statusy_id" +
37         " where s1.morphosyntactic>=13 and
38         s2.morphosyntactic<=12 )"
39     ).
40     each { |at211|
41         puts "poprawiam status #{at211.akapit_id}/#{at211.id}"
42         at211.set_status( :morphosyntactic, :dopuszczony, :tylko_to )
43         sth_done = true unless sth_done
44     }
45
46     puts " »sth. done« is #{sth_done.inspect}"
47 end
48
49 def self.down
50 end
51 end

```

#### **o43\_popraw\_seg2946.rb**

```

23 class PoprawSegm2946 < ActiveRecord::Migration
24     extend MigrationHelper
25
26     def self.up
27         if false
28             [
29                 ['-7', 201447, true, 529, 1075 ],
30                 ['- ', 279451, false, 529, 1076 ],
31                 ['7', 279452, false, 529, 1076 ],
32                 [',', 201448, true, 529, 1075 ],
33                 ['16zł', 201449, false, 454, 923],
34                 ['16', 279357, true, 529, 1075 ],
35                 ['zł', 279358, true, 454, 924 ]
36             ].each { |arr212|
37                 t212 = Token.find( arr212[1] )
38                 t212.chosen, t212.sg_choice_id, t212.sg_variant_id = arr212[2,
39                     3]
40                 t212.save!
41             }
42
43             # A teraz na podstawie takiej segmentacji należy zgłosić prośbę o nową
44             segmentację.
45         end
46     end
47
48     def self.down
49     end
50 end

```

**o44\_token\_index\_orth.rb**

```
23  class TokenIndexOrth < ActiveRecord::Migration
24
25    extend MigrationHelper
26
27    def self.up
28      kill_anotatorni _if_8003
29
30      addindex :token, [ :akapit_id, :orth ]
31      addindex :token, :orth
32    end
33
34    def self.down
35    end
36  end
```

**o45\_popraw\_intrj.rb**

```
23  class PoprawIntrj < ActiveRecord::Migration
24
25    extend MigrationHelper
26
27    def self.up
28      if AnoVersion.nkjp?
29        kill_anotatorni _if_8003
30
31        # poprawiamy intrj na interj, co miało być zrobione dawno, dawno
32        temu.
33
34        if Leksem.find_by_klasa_gram_id( 12 )
35          execute 'update leksem set klasa_gram_id=38 where
36                  klasa_gram_id=12'
37        end
38      end
39    end
40  end
41  end
```

**o46\_popraw\_sierzanty.rb**

```
23  class PoprawSierzanty < ActiveRecord::Migration
24
25    extend MigrationHelper
26
27    def self.popraw( tabela, kolumna, stara_wartość, nowa_wartość )
28      execute "update #{tabela} set #{kolumna}= '#{nowa_wartość}' where
29              #{kolumna}='#{stara_wartość}' "
30    end
31
32    def self.up
33      kill_anotatorni _if_8003
```

```

35     self.popraw :token, :orth, '>', '&gt;';
36     self.popraw :token, :orth, '<', '&lt;';
37     self.popraw :token, :orth, '&', '&amp;';
38     ##      execute "update token set orth= '&gt;'" where orth='>' "
40     ##      execute "update token set orth= '&lt;'" where orth='<' "
42     ##      execute "update token set orth= '&'" where orth='&' "
44     self.popraw :leksem, :lemat, '>', '&gt;';
45     self.popraw :leksem, :lemat, '<', '&lt;';
46     self.popraw :leksem, :lemat, '&', '&amp;';

48     end

51     def self.down
52     end
53 end

```

#### o47\_popraw\_interpretacja\_leksem.rb

```

23 class PoprawInterpretacjaLeksem < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28
29         execute "analyze"
30         execute "vacuum"
31
32         ##      addindex :interpretacja, :path_id
33         ##      addindex :leksem, :path_id
34
35         if Interpretacja.find_by_sql(
36             " select * from interpretacja i inner join
37               leksem l using( leksem_id )" +
38             "where i.path_id<>l.path_id limit 1" )[0]
39             # poprawiamy przypisanie leksemów z nie tej ścieżki
40
41             kill_anotatorni _if_8003
42
43             # na wypadek, gdyby ta poprawka okazała się błędem, zapisujemy co
44             # było przed nią.
45             execute "create table if not exists
46               interpretacja_bef47_bad_leksem( " +
47                 " interpretacja_id integer primary key, " +
48                 " leksem_id integer );"
49
50             Interpretacja.find_all_by_dodana( true ).each { |i218|
51                 le218 = i218.leksem
52
53                 conds = {
54                     :lemat => le218.lemat,
55                     :path_id => i218.path_id,
56                     :klasa_gram_id => le218.klasa_gram_id
57                 }
58
59                 le219 = Leksem.find( :first, :conditions => conds )
60                 le219 ||= Leksem.create!( conds.update(

```

```

63                                     :lex_xmlid => i218.lex_xmlid
64                                     ))
65     if le218 != le219
66
67         execute "insert into interpretacja_bef47_bad_leksem " +
68             " values( #{i218.id}, #{i218.leksem_id} )"
69
70         i218.leksem = le219
71         i218.save!
72         ##             puts "#{i218.path_id} #{i218.id}"
73     end
74 }
75
76 end
77
78 if Interpretacja.find_by_sql(
79     " select * from interpretacja i1 inner
80     join interpretacja i2 " +
81     " using( token_id, leksem_id ) " +
82     " where i1.lex_xmlid<>i2.lex_xmlid limit 1
83     " )[0]
84
85     # poprawiamy zawyżony (niepotrzebnie unikatowy) lex_xmlid
86
87     # na wypadek, gdyby ta poprawka okazała się błędem, zapisujemy co
88     # było przed nią.
89     execute "create table if not exists
90     interpretacja_bef47_bad_xmlids( " +
91     " interpretacja_id integer primary key, " +
92     " lex_xmlid text, " +
93     " msd_xmlid text, " +
94     " numer_lex_token integer );"
95
96     Interpretacja.find_all_by_dodana( true ).each { |i218|
97
98         # szukamy oryginalnej interpretacji z naszego lexa.
99         conds218 = " token_id = #{i218.token_id} and leksem_id =
100         #{i218.leksem_id} and interpretacja_id<>#{i218.id} "
101
102         i219 = Interpretacja.find( :first, :conditions => conds218 )
103         if i219 and
104             i219.lex_xmlid != i218.lex_xmlid
105
106             execute "insert into interpretacja_bef47_bad_xmlids " +
107                 " values( #{i218.id}, '#{i218.lex_xmlid}', " +
108                 " '#{i218.msd_xmlid}', #{i218.numer_lex_token} ) "
109
110             # poprawiamy też msd_xmlid
111
112             msd_id = Interpretacja.find( :all, :conditions => conds218 ).
113                 collect { |il| il.msd_xmlid.split('.')[0].to_i }.
114                 compact.max + 1
115
116             ##             puts "#{i218.path_id} #{i218.id}"
117
118             i218.lex_xmlid = i219.lex_xmlid
119             i218.msd_xmlid = "#{i219.lex_xmlid}.#{msd_id}"

```

```

117         i218.numer_lex_token = msd_id
118         i218.save!
119     end
120 }
121 end
124 end # of self.up
126 def self.down
127     end
128 end

```

#### 048\_nie\_przypisane\_sensy.rb

```

23 class NiePrzypisaneSensy < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28         kill_anotatorni _if_8003
29
30         CzMLeksem.przypisz_leksemom_i_interpretacjom
31
32         # Spytałem AP, co lepiej zrobić. Stwierdził, że lepiej
33         Statusy.find_by_sql(
34             "select * from statusy where word_senses>=16 and
              exists(select * from interpretacja i inner join
              token t on i.token_id=t.token_id and
              i.disamb='t'where t.akapit_id=statusy.akapit_idand
              i.cz_m_leksem_id is not null andnot exists (select
              * from sens_anot sa where sa.token_id=t.token_id
              andsa.akapit_transzy_id=statusy.akapit_transzy_id));
              " ).each { |sta|
35             puts "#{sta.id}  #{sta.transza_id}"
36             sta.set_poziom( :word_senses, :dopuszczony )
37         }
38
39     end
40
41     def self.down
42         end
43 end

```

#### 049\_popraw\_polpauze.rb

```

23 class PoprawPolpauze < ActiveRecord::Migration
24
25     extend MigrationHelper
26
27     def self.up
28         kill_anotatorni _if_8003
29
30         Token.find( :all, :conditions => " xpointer like '%-%' " ).each {
31             |t222|
32             puts t222.id
33
34             t222.xpointer = t222.xpointer.gsub( '-', '-' )

```

```

35         t222.save!
36     }

38     ##      execute " update token set xpointer=replace( xpointer, '-',
39     ' ' ) " +
40     ##      " where xpointer like '%-%' "
41     # na Chopinie nie ma replace
42     end
43
44     def self.down
45     end
46
47     end

```

### **o5o\_popraw\_xpointery.rb**

```

23     class PoprawXpointery < ActiveRecord::Migration
24
25         extend MigrationHelper
26
27         def self.up
28             if AnoVersion.nkjp?
29                 kill_annotatorni _if_8003
30
31                 addindex :token, :dodany
32
33                 execute "analyze"
34
35                 SgVariant.tokododane.each { |sgv|
36                     unless [ 2513, 2528 ].include?( sgv.id ) # ręcznie oporządzane
37                         »8,3kg« i »5,5kg«. Poprawimy ręcznie dalej.
38                         sgv.popraw_xpointers
39                     end
40                 }
41
42                 # ręcznie zmodyfikowany sgv. 2513
43                 Token.ustaw_xpointer_na( 553380,
44                     "string-range(txt_1.1-ab,47,3)" )
45                 Token.ustaw_xpointer_na( 553366,
46                     "string-range(txt_1.1-ab,50,2)" )
47                 # razem konsumują 6 znaków (ze spacją) tekstu wejściowego
48
49                 # ręcznie zmodyfikowany sgv. 2528
50                 Token.ustaw_xpointer_na( 553393,
51                     "string-range(txt_1.1-ab,70,3)" ) # 5,5
52                 Token.ustaw_xpointer_na( 553364,
53                     "string-range(txt_1.1-ab,73,3)" ) # kg
54                 # razem konsumują 6 znaków tekstu wejściowego.
55
56                 end
57             end # of self.up
58
59             def self.down
60             end
61
62             end

```

**051\_add\_interpretacja\_export.rb**

```

23  class AddInterpretacjaExport < ActiveRecord::Migration
24    extend MigrationHelper
25
26    def self.up
27      kill_anotatorni _if_8003
28
29      ##      execute "drop view if exists msd_opis"
30
31      ##      execute \"create view msd_opis as
32      ## select leksem_id, lemat, klasa_gram_ozn
33      ## from leksem inner join klasa_gram
34      ## using( klasa_gram_id ); \"
35
36      execute " drop view if exists interpretacja_export"
37
38      execute " create view interpretacja_export asselect
39      interpretacja_id as interpretacja_export_id,interpretacja.*,lemat,
40      klasa_gram_oznfrom interpretacja inner join leksem using(
41      leksem_id )inner join klasa_gram using( klasa_gram_id )where
42      interpretacja.dodana='f' or interpretacja.disamb='t'order by
43      numer_lex_token, interpretacja_id"
44      # porządkowanie wg ID jest bazodanowo niekoszerne, ale lepiej tak
45      (2009/7/24)niż dodawać kolumnę z numerkiem wyłuskany
46      z msd_xmlid i pieczołowicie pilnować, by zawsze się dobrze łuskało.
47      # Proszę zwrócić uwagę, że nie eksportujemy tagów dodanych ręcznie,
48      o ile nie zostały wybrane jako dezambiguacja.
49
50    end
51
52    def self.down
53    end
54  end

```

**052\_popraw\_segmn10391.rb**

```

23  class PoprawSegm10391 < ActiveRecord::Migration
24    extend MigrationHelper
25
26    def self.up
27      if AnoVersion.nkjp?
28        kill_anotatorni _if_8003
29
30        pa = ProsbyAnotatorek.new(
31          :uzytkownik_id => Uzytkownik.natrnr.id,
32          :opis => "#346, zgł. przez interpreting",
33          :dotyczy =>
34            NowaSegmentacja.new(
35              :ids => [ 10391 ],
36              :total_akapitow => 17,
37              :seg_is => [
38                "31października" ],

```



```

39             :seg_should_be =>
40             %w( 31 października ),
41             :nps => false, # separated
42             with a space
43             :wprowadzona => false,
44             :akapit_transzy_id =>
45             20784,
46             :transza_id => 2079
47         )
48     end
49     pa.save!
50 end
51 end # of up
52
53 def self.down
54 end
55 end
56 end

```

### 053\_xpointers\_not\_null.rb

```

23 class XpointersNotNull < ActiveRecord::Migration
24   extend MigrationHelper
25
26   def self.up
27     if AnoVersion.nkjp?
28       Token.popraw_xpointery_nullowe
29       './anotatornia-mongrels stop'
30     end
31   end
32
33   def self.down
34   end
35 end
36
37 def self.down
38 end

```

### 054\_popraw\_dziabag\_ta.rb

```

23 class PoprawDziabagTa < ActiveRecord::Migration
24   def self.up
25     pa = ProsbyAnotatorek.find( :first, :conditions => {
26       :prosby_anotatorek_id => 2201,
27       :dotyczy_id => 2131,
28       :dotyczy_type => "NowaSegmentacja",
29       :uzytkownik_id => 59,
30       :updated_at => '2010-04-19 21:04:18'
31     }
32   )
33   puts "\n\n\n@@@@@nni ma takiej\n@@@@@n\n" unless pa
34
35   if pa
36     pado = pa.dotyczy
37     pado.wprowadzona = false

```

```
38     pa.rozpatrzona = false
39     pa.spelniona = nil
40     pado.save!
41     pa.save!
43   end # of if pa
45 end
47 def self.down
48   end
49 end
```

### **055\_popraw\_sg\_nom.rb**

```
23 class PoprawSgNom < ActiveRecord::Migration
25   extend MigrationHelper
28   def self.up
29     kill_annotatorni _if_8003_nobackup
31     # We use the fact that the tagset of the Corpus consists only of ASCII
      characters.
32     execute 'update interpretacja set reszta_tagu=lower(reszta_tagu),
      ' +
33       ' updated_at = current_timestamp ' +
34       ' where reszta_tagu<>lower(reszta_tagu) '
35   end
37   def self.down
38     end
39 end
```