

# TermoPL<sup>\*</sup> 8.0

## user manual

Institute of Computer Science  
Polish Academy of Sciences

30.12.2024

---

<sup>\*</sup>TermoPL is co-funded by CLARIN-PL

# 1 Introduction

TermoPL is a tool created to extract terminology from domain corpora. It can also be used for other languages as long as you define the appropriate tagset and grammar. The program extracts phrases, candidates for terms, using Universal Dependency (UD) structures obtained from UD parsers or through a simple grammar that can be customized. It applies the C-value method to rank term candidates being either the longest identified acceptable phrases or their nested subphrases. The method operates on simplified base forms in order to unify morphological variants of terms and to recognise their contexts. For the method using simple grammar templates, we support the recognition of nested terms by word connection strength which allows us to eliminate truncated phrases from the top of the term list. For Polish, the program has an option to convert simplified forms of phrases into correct phrases in the nominative case. TermoPL accepts as input morphologically annotated and disambiguated domain texts and creates a list of terms, the top part of which comprises domain terminology. It can be used to compare two candidate term lists using four different coefficients showing asymmetry of term occurrences in this data. For Polish texts, TermoPL can group semantically related terms using plWordNet<sup>1</sup>.

You can learn more about TermoPL from [7].

## 2 How it works?

ThermoPL can extract candidate phrases for terms using two methods. In the first, terms are identified using simple grammar templates that can be customised. In the second, terms are extracted from universal dependency (UD) trees created by the UD parser.

Regardless of which mode the program operates in, terms are identified by their simplified (lemmatised) forms. Simplified forms enable the program to recognise all morphological forms of a phrase as corresponding to one term. Morphological forms of phrases may significantly differ for languages with reach inflection such as Polish. For example, *katedra romańska* ‘romanesque cathedral’ whose simplified form is *katedra romański* has 14 forms (e.g. *katedrze romańskie<sub>loc,sg</sub>*, *katedrom romańskim<sub>dat,pl</sub>*) depending on the case and number.

The number of considered term candidates can be reduced by the user, if he/she submits a list of lemmas of stop words. If a term candidate contains any of the stop words, it is eliminated. For example, *ta katedra romańska* ‘this romanesque cathedral’ should be excluded from the list of term candidates for obvious reasons, although it conforms to the grammar used by the program. Similar problems produce compound prepositions. For example, the compound preposition *z naszego punktu widzenia* ‘from our point of view’ contains the grammatically valid term candidate *nasz punkt widzenia* ‘our point of view’, which should not be considered as a term. One can further shrink the list of considered terms, if he/she specifies the list of general or out-of-domain terms.

Once the list of phrases containing potential terms has been determined, it is sorted according to the value of the term ranking function from highest to lowest. As the ranking function the C-value is used (see Section 3.1).

Finally, the program attempts to rewrite all terms from their simplified (lemmatised) forms to their basic (nominative) forms. A base form of a term is usually singular, unless all phrases (maximal or nested) corresponding to this term are plural noun phrases. Letter case used in base forms is determined by orthographic forms associated with each term. If a particular word appears in upper case in all phrases, it remains in upper case in the base form. Otherwise, it is converted to lower case. For Polish, the new version of the morphosyntactic analyser and

---

<sup>1</sup><http://plwordnet.pwr.wroc.pl/wordnet/>

generator Morfeusz [14] is used in this process. For other languages, TermoPL prefers the nominative singular form or the most common form of the term if its nominative form is not present.

A generated list can be truncated by the user to include only multi-word terms and/or some specified number of top ranked term candidates.

The program can be used in two modes: batch and interactive. For the interactive mode a graphical user interface is provided.

## 2.1 Acceptable input

The program accepts UTF8 encoded input with morphosyntactic analysis or additionally with grammatical structure in various formats such as NKJP[10], XCES[5], CoNLL-U<sup>2</sup> and the simple, flat format in which each token is represented by a single line of text consisting of an orthographic **form** (as it appears in a processed document), its **lemma** and a **tag**. The following two lines are acceptable input describing the token in flat format files:

```
form<TAB>#lemma<TAB>#tag#
form<TAB>lemma<TAB>tag
```

Sentences are separated by an empty line or one of the lines below:

```
&<TAB>##<TAB>##
&<TAB>&<TAB>&
```

The input file of the flat format may contain more than one document from the analysed corpus, which are separated by a line of text starting with **%**. Separating documents is useful if we want to compare corpora using the term weight method described in Section 3.6.

For backward compatibility, the program also supports files in which each token is represented as follows:

```
form<TAB>lemma<TAB>tag, or
form<TAB>lemma<TAB>tag<TAB>nps,
```

where 'nps' means 'no preceding space' and sentences are separated by a line containing single string 'eos' (end of sentence). Files in this format must have the extension 'tgt'.

It is also possible to use untagged input files. In this case, the files are first processed by the Stanza Dependency Parser [11], which creates files in CoNLLu format.

## 2.2 Using simple template grammars

TermoPL reads input sentence by sentence and identifies the maximal sequences of consecutive tokens that are recognised, either by the standard built-in grammar presented in Figure 1, or a custom grammar provided by the user. In the built-in grammar, *NAP* and *NAP\_GEN* both denote noun phrases, with the proviso that *NAP\_GEN* denotes noun phrases in the genitive case. It is assumed, of course, that tokens matched by *NAP* (and *NAP\_GEN*) must agree in number, case and gender. In other words, the program first extracts the longest (maximal) phrases consisting of a noun phrase, possibly modified by other noun phrases in the genitive case. Then, it splits them into smaller parts (nested phrases) that still conform to the given grammar. It provides four methods for splitting maximal phrases. The first one searches for all subphrases that satisfy the given grammar. This method produces considerably more term candidates than the remaining three methods, since it does not care if the resulting terms are

<sup>2</sup><https://universaldependencies.org/format.html>

```

NPP : $NAP NAP_GEN*;
NAP[agreement] : AP* N AP*;
NAP_GEN[case = gen] : NAP;
AP : ADJ | ADJA DASH ADJ | PPAS;
^N[pos = subst, ger];
ADJ[pos = adj];
ADJA[pos = adja];
PPAS[pos = ppas];
DASH[form = "-"];

```

Figure 1: The built-in grammar.

semantically odd, truncated phrases. For example the phrase *nominalna roczna stopa procentowa* 'nominal annual interest rate' contains a grammatically acceptable subphrase *roczna stopa* which looks odd and should not be accepted as a term. The rest of the phrase splitting methods try to eliminate such phrases using NPMI driven recognition of nested phrases (see Section 3.2) introduced in [6]. These methods try to split a phrase at the weakest connection point expressed by NPMI coefficient. The first method always divides the phrase at the weakest connection, regardless whether the resulting subphrases conform to the given grammar. The second one tries to divide the phrase into subphrases so as to at least one of them satisfies the grammar rules. The third method is very similar to the second one except that it prefers cases when two of the resulting subphrases satisfy the grammar. This preference is expressed by some predefined coefficient. By default, TernoPL sets this coefficient to 120%.

To obtain base forms a token or a group of tokens matched with a symbol marked with the \$ character are replaced by their nominative forms. All other tokens are left unmodified. In the grammar given above, the only symbol marked with \$ is *NAP*. Therefore all *NAP* phrases are transformed into their nominative forms, whereas *NAP\_GEN* phrases are left as they appear.

It is necessary to indicate the head phrase of the extracted term. The symbol identifying the phrase head is preceded by a ^ character. In the built-in grammar, the N symbol (noun) is chosen as the head of the phrase.

## 2.3 Using universal dependency structures

The UD project assumes a consistent structure of annotation schemes for many languages. In the terminology candidate identification algorithm described below, this consistency is used to define rules for selecting nominal phrases that are based on four sets of information. Two sets consist of UD POSs. The first – **head-pos** – contains UD POSs of nodes that can be heads of the term phrases, i.e. **NOUN** and **PROPN**. The second – **non-head-pos** – contains UD POSs of nodes that can be part of the term phrases but not their heads, i.e. **ADJ**, **ADP**, **ADV**, **DET** and **NUM**. The next two sets consist of relations: **obligatory-rel** and **facultative-rel**. The first set groups relations between words that should appear together in terminology phrases, while the second set contains relations between words that may or may not appear in a sentence. The appropriate relations are listed below:

```

obligatory-rel: amod:flat, case, case:poss, ccomp, compound, compound:prt, det, expl:pv, fixed,
flat, iobj, nmod:arg, nmod:flat, nsubj:ger, obj, obl:agent,obl:arg, xcomp.

```

**facultative-rel:** acl, advmod, advmod:emph, amod, appos, nmod, nmod:poss, nummod, nummod:gov, obl

First, the program selects all the nodes that can be included in the terminology phrases, creating a list of potential terminology nodes. This list includes all nodes whose UD POS belong to one of the above-mentioned sets: **head-pos** or **non-head-pos**. For hyphenated compound words, which are allowed in many languages, all nodes of the UD structures representing them are placed in the list of potential terminology nodes. All nodes from structures representing hyphenated compound words, except those that are heads of these structures, are also placed in the list of hyphenated nodes. Each of the nodes in the list of hyphenated nodes will be selected for creating phrases if and only if its head is also selected.

In the structure where only relations between terminology nodes are left in place, it is checked whether, for each node, all **obligatory-relations** are in the current structure. Doing so, some truncated phrases are avoided, as we do not want to create phrases with nodes that have unrealized requirements.

The process of making phrases is repeated in a loop for all nodes in the list of potential terminology nodes. For each node, all combinations of dependent nodes are considered, where nodes connected by **obligatory-relations** and those from the list of hyphenated nodes must be included in the phrase, while nodes connected by **facultative-relations** may be omitted. Only those phrases for which the head element is in the **head-pos** set are accepted as term candidates. The list of established phrases for the considered node is passed to the upper node (if relevant), and the considered node is removed from the list of potential terminology nodes. The whole procedure is repeated until the list of potential terminology nodes is empty.

More on the extraction of terms directly from UD structures can be found in [8].

### 3 Formulas used in calculations

Let us first introduce some useful notations:

$A$	domain corpus
$B$	contrastive corpus
$AB$	merged corpora $A$ and $B$
$T(X)$	set of terms of a corpus $X$
$D(X)$	set of documents in a corpus $X$
$t$	term
$d$	document
$f_X(t)$	frequency of a term $t$ in a corpus $X$
$f_t(d)$	frequency of a term $t$ in a document $d$
$N_X^Y$	size of a corpus $X$ with respect to $T(Y)$ , i.e. $\sum_{t \in T(Y)} f_X(t)$
$S_X$	size of a corpus $X$ , i.e. $N_X^X$

#### 3.1 C-value

TermoPL ranks term candidates using modified version of C-value described in [4]. For a given phrase  $p$ , its C-value is defined as follows:

$$C\text{-value}(p) = \begin{cases} l(p) \times \left( f(p) - \frac{1}{|LP|} \sum_{lp \in LP} f(lp) \right), & \text{if } |LP| > 0, \\ l(p) \times f(p), & \text{if } |LP| = 0, \end{cases}$$

where  $f(p)$  is the number of occurrences of a phrase  $p$ ,  $LP$  is a set of different phrases containing  $p$ ,  $|LP|$  is the number of phrases in  $LP$  and  $l$  is a function which increases weight for longer

phrases. It is equal to the logarithm ( $\log_2$ ) of phrase length for multi-word expressions and a constant (TermoPL uses 0.1) for one-word terms.

What it follows from the above equation, the chance that a given phrase can be assumed as a domain term increases with the number of contexts in which it occurs. What is meant by a context and hence, what the term "different phrases" means in the definition of C-value will be explained in Section 6.5 (page 17).

### 3.2 Normalised pointwise mutual information

In order to determine the connection strength for a pair of words, TermoPL counts normalised pointwise mutual information (NPMI) proposed by [2] for all lemmatized bigrams in a considered corpus.

$$NPMI(x, y) = \left( \ln \frac{p(x, y)}{p(x)p(y)} \right) / -\ln p(x, y),$$

where  $p(x, y)$  is a probability of the ' $x$   $y$ ' bigram in the considered corpus, and  $p(x)$ ,  $p(y)$  are probabilities of ' $x$ ' and ' $y$ ' unigrams, respectively.

### 3.3 Corpora-comparing log-likelihood

The Corpora-comparing log-likelihood (LL) coefficient [12] points out whether or not a given term occurs significantly more frequent in one of two tested corpora. It is calculated in the following way:

$$LL(t) = 2 \left( f_A(t) \log \left( \frac{f_A(t)}{E_A(t)} \right) + f_B(t) \log \left( \frac{f_B(t)}{E_B(t)} \right) \right),$$

where  $E_X = S_X \frac{f_A(t) + f_B(t)}{S_A + S_B}$ . In calculations we can use C-values instead of frequencies. The size  $S_X$  of a corpus is measured then by the sum of C-values of all its terms.

### 3.4 Term frequency inverse term frequency

Term frequency inverse term frequency (TFITF) method [1] combines the frequency of a term  $t$  in the domain corpus with inverse term frequency in both domain and contrastive corpora.

$$TFITF(t) = \log(f_A(t)) \log \left( \frac{N_{AB}^A}{f_{AB}(t)} \right).$$

We can choose to use C-values instead of frequencies in all calculations, just as in case of LL coefficient.

### 3.5 Contrastive selection of multi-word terms

Contrastive selection of multi-word terms (CSmw) [1] is defined by the following equation:

$$CSmw(t) = \log \left( \log(f_A(t)) \times N_B^A \times \frac{f_A(t)}{f_B(t)} \right)$$

CSmw coefficient can also be calculated with C-values.

### 3.6 Term weight

Term weight (TW) [9] depends on the domain relevance (DR) of a term  $t$  and its domain consensus (DC) expressed by the entropy of the distribution of  $t$  in the domain corpus  $A$ .

$$TW(t) = \alpha DR(t) + \beta DC^*(t),$$

where  $\alpha$  and  $\beta$  are numbers from  $(0, 1)$ , and  $DR$  and  $DC$  are defined as follows:

$$\begin{aligned} DR(t) &= \frac{P_A(t)}{\max(P_A(t), P_B(t))}, \\ P_X(t) &= \frac{f_X(t)}{S_X}, \\ DC(t) &= - \sum_{d \in D(A)} (p_t(d) \log(p_t(d))), \\ DC^*(t) &= \frac{1}{L} DC(t), \\ L &= \max_{t \in T(A)} DC(t), \\ p_t(d) &= \frac{f_t(d)}{S_A}. \end{aligned}$$

Unlike the other three methods presented above, TW works on frequencies only. Default values for  $\alpha$  and  $\beta$  are 0.9 and 0.3, respectively.

## 4 Customising the tagset

TermoPL allows the use of alternative tagsets to define grammars.

The **tag** consists of a list of morphosyntactic markers. The first element of this list is always corresponds to the grammatical class (**pos**) of the segment. It is followed by markers defining grammatical categories that characterise the selected segment.

$$\text{pos} : \text{cat}_0 : \dots : \text{cat}_n, \quad n \geq 0.$$

Sometimes the tags are reduced to just a part of speech. Part-of-speech tags are used in the Penn Treebank Project<sup>3</sup>, for example. In order to use this kind of tagset, one has to put the following line as the only line in the tagset structure definition file:

$$\text{TAG} = \text{pos}.$$

By default, list items that make up a tag are separated by a colon, but the user can change this using **DELIMITER** directive in a tagset structure definition file. For example, if we decide to separate items with comma, we have to put the following line as the first line in a tagset structure definition file:

$$\text{DELIMITER} = ", ".$$

---

<sup>3</sup>[https://www.ling.upenn.edu/courses/Fall\\_2003/ling001/penn\\_treebank\\_pos.html](https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)

Sometimes the tags are fixed-length strings where each position encodes one morphological category with one character. Such positional tags are used, for example, by the Prague Dependency Treebank 2.0<sup>4</sup>. In such cases we use an empty separator:

DELIMITER = "".

TermoPL allows you to define tagset structure by two methods. The first of them (**categories by positions**) requires that for a given grammar class, markers defining grammatical categories always appear in the same order in the tag. Using this method of defining a tagset, we do not need to specify what values the markers corresponding to particular grammatical categories can take. The program will not check whether a given value actually corresponds to a given category. The user must define the set of grammatical categories of the tagset and the order of occurrence of markers corresponding to the categories in the tag.

```
<categories by positions>
  subst: number, case, gender, sgender
  adj: number, case, gender, degree
  ...
```

It may happen that a certain grammatical category does not apply to the description of the segment, although in the general case it characterises the class to which the segment belongs. Such a category is, among others, **sgender** for class **subst** in the default tagset used by TermoPL. The marker (let's call it **cat<sub>i</sub>**) corresponding to this category can be omitted from the tag, but only if it appears at the end. Otherwise, it can be substituted by some placeholder or an empty space must be left at this position in the tag, where, according to the class definition, this marker should appear:

**pos : cat<sub>0</sub> : ... : cat<sub>i-1</sub> : : cat<sub>i+1</sub> : ... : cat<sub>n</sub>.**

For positional tags such as those used by the Prague Dependency Treebank, the user must specify grammatical categories and their order in the tag.

```
<categories by positions>
  cat0, cat1, ..., catn
```

The second method (**categories by values**) assumes that the tag values corresponding to different grammatical categories are different. The grammatical category will be identified in this case based on the marker value. When defining grammatical categories, the user must specify the values of the corresponding markers.

```
<categories by values>
  number: sg, pl
  case: nom, gen, dat, acc, inst, loc, voc
  gender: m1, m2, m3, f, n
  sgender: ncol, col
  degree: pos, com, sup
  ...
```

The default TermoPL tagset is defined by the **categories by values** method.

The user can define composite categories whose values will be sets containing the values of other basic categories defined in the tagset.

---

<sup>4</sup><http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/m-layer/html/ch02.html>



```
<definitions>
  agreement = number, case, gender
  ...
```

TermoPL is using the following default tagset:

```
DELIMITER = ":"
<categories by values>
  number: sg, pl
  case: nom, gen, dat, acc, inst, loc, voc
  gender: m1, m2, m3, f, n
  sgender: pt, ncol, col
  person: pri, sec, ter
  degree: pos, com, sup
  aspect: imperf, perf
  negation: aff, neg
  accent: akc, nakc
  pprep: praep, npraep
  accom: congr, rec
  aggl: nagl, agl
  vocality: wok, nwok
  fstop: pun, npun
```

```
<definitions>
  agreement = number, case, gender
```

class	meaning (in Polish)	categories
<b>fin</b>	forma nieprzeszła	number:person:aspect
<b>bedzie</b>	forma przyszła czasownika BYĆ	number:person:aspect
<b>aglt</b>	aglutynant czasownika BYĆ	number:person:aspect:vocality
<b>praet</b>	pseudomiesłów	number:person:gender:aspect:aggl
<b>impt</b>	rozkaznik	number:person:aspect
<b>imps</b>	bezosobnik	aspect
<b>inf</b>	bezokolicznik	aspect
<b>pcon</b>	imiesłów przysłówkowy współczesny	aspect
<b>pant</b>	imiesłów przysłówkowy uprzedni	aspect
<b>ger</b>	odśownik	number:case:gender:aspect:negation
<b>pact</b>	imiesłów przymiotnikowy czynny	number:case:gender:aspect:negation
<b>ppas</b>	imiesłów przymiotnikowy bierny	number:case:gender:aspect:negation
<b>winien</b>	czasownik typu WINIEN	number:gender:aspect
<b>pred</b>	predykatyw	—
<b>subst</b>	rzeczownik	number:case:gender:sgender
<b>depr</b>	rzeczownik – forma deprecjatywna	number:case:gender
<b>adj</b>	przymiotnik	number:case:gender:degree
<b>adja</b>	przymiotnik przyprzymiotnikowy	—
<b>adjp</b>	forma poprzyimkowa	case
<b>adjc</b>	przymiotnik predykatywny	—
<b>adv</b>	przysłówek	degree
<b>num</b>	liczebnik	number:case:gender:accom:sgender
<b>ppron12</b>	zaimek nietrzecioosobowy	number:case:gender:person:accent

continued...

class	meaning (in Polish)	categories
<b>ppron3</b>	zaimek trzecioosobowy	number:case:gender:person:accent:pprep
<b>siebie</b>	zaimek SIEBIE	case
<b>prep</b>	przyimek	case:vocality
<b>conj</b>	spójnik współrzędny	—
<b>comp</b>	spójnik podrzędny	—
<b>brev</b>	skrót	fstop
<b>interj</b>	wykrzyknik	—
<b>part</b>	partykuła	vocality
<b>frag</b>	człon frazeologizmu	—
<b>interp</b>	interpunkcja	—
<b>ign</b>	forma nierozpoznana	—

Table 2: Grammatical classes in the default tagset

category	meaning (in Polish)	values
<b>number</b>	liczba	sg, pl
<b>case</b>	przypadek	nom, gen, dat, acc, inst, loc, voc
<b>gender</b>	rodzaj	m1, m2, m3, f, n
<b>sgender</b>	przyrodzaj	pt, ncol, col
<b>person</b>	osoba	pri, sec, ter
<b>degree</b>	stopień	pos, com, sup
<b>aspect</b>	aspekt	imperf, perf
<b>negation</b>	zanegowanie	aff, neg
<b>accent</b>	akcentowość	akc, nakc
<b>pprep</b>	poprzyimkowość	praep, npraep
<b>accom</b>	akomodacyjność	congr, rec
<b>aggl</b>	aglutynacyjność	nagl, agl
<b>vocality</b>	wokaliczność	wok, nwok
<b>fstop</b>	kropkowność	pun, npun

Table 3: Grammatical categories in the default tagset

## 5 Customising the grammar

As it was mentioned, the built-in grammar can be replaced by some user-defined grammar. To specify a grammar one has to define production rules and tests that have to be performed on tokens or sequences of tokens during the matching process. Rules have the following form:

```
<symbol> [ "[" <test-list> "]" ] ":" <regular expression over symbols> ";",
<symbol> "[" <test-list> "];".
```

The left-hand side of a rule consists of only one nonterminal symbol. The right-hand side is a regular expression over the set of symbols. Regular expressions allowed by the program may contain alternatives separated by '|', and quantifiers: '?', '\*' and '+', which indicate zero or one, zero or more and one or more occurrences of the preceding symbol, respectively. No loops

are allowed, which means that the rewriting process cannot yield to symbol that appeared on the left hand-side of an applied rule.

For each symbol it is possible to specify `<test-list>`, i.e. a test or a series of tests performed during the matching process.

`<test> [ ", " <test> ]`

Tests can be defined on the left-hand side of a rule or in separate statements. Tests, separated with semicolons, are placed in square brackets just after a symbol to which they relate.

A test can be an agreement checking function that refers to some grammatical category. For a given sequence of tokens, it returns true if and only if all tokens sharing the grammatical category being tested are assigned the same value for this category. A test can also be an expression returning boolean value:

`<selector> <op> <string> [", " <string> ],`

where **selector** is a function defined on tokens and lists of tokens and returning a string value, and **op** is one of the following operators: `'='`, `'!='`, `'~'` and `'! ~'`. The first two operators serve to compare strings if they are equal (`'='`) or not (`'!='`). With the remaining operators we can check whether a string returned by a selector matches (`'~'`) or not (`'! ~'`) a Java-style regular expression. If there are more strings on the right side of a positive operator (`'='` or `'~'`), a test succeeds whenever it succeeds for at least one of these strings. In case of negative operators (`'!='` or `'! ~'`) a test succeeds if it succeeds for all given strings.

Tests can be applied to single tokens or sequences of tokens. In the built-in grammar presented on page 3,  $N[pos = subst, ger]$  means that a token matched with symbol  $N$  must be a substantive or a gerund, whereas  $NAP[agreement]$  means that a sequence of tokens matched with  $NAP$  must agree in number, case and gender, since *agreement* is a composite category consisting of categories *number*, *case* and *gender* defined in the tagset (see page 8). The expression  $NAP[agreement]$  can be replaced by  $NAP[number; case; gender]$ . Note that a sequence of tokens matched with  $NAP$  may contain tokens for which *agreement* test is not applicable, e.g. `'-'`. In such cases testing is performed only on those tokens for which it makes sense.

ThermoPL provides four built-in selectors whose names are self-explaining: *form*, *lemma*, *tag* and *pos*. The other selectors correspond to the grammatical categories defined in the tagset.

## 6 Graphical user interface

ThermoPL is a multi-document application. The user may open and analyse several sets of terms simultaneously. The graphical user interface of the program consists of several windows, dialog boxes and menus. In case of Windows and Unix operating systems, all these graphical elements are gathered in one virtual desktop window.

The user can navigate through all functionality of the program using menus. They are located either on the top of the screen (Mac OSX), or on the top of the virtual desktop window (Windows, Linux).

### 6.1 The Terms Window

When the program finishes the extraction process it displays a table of term candidates as it is shown in Figure 9. For every term the table shows: term's position on the list (**#**), its rank (**Rank**), base/simplified form (**Term**), C-value (**C-value**), length (**Length**), number of

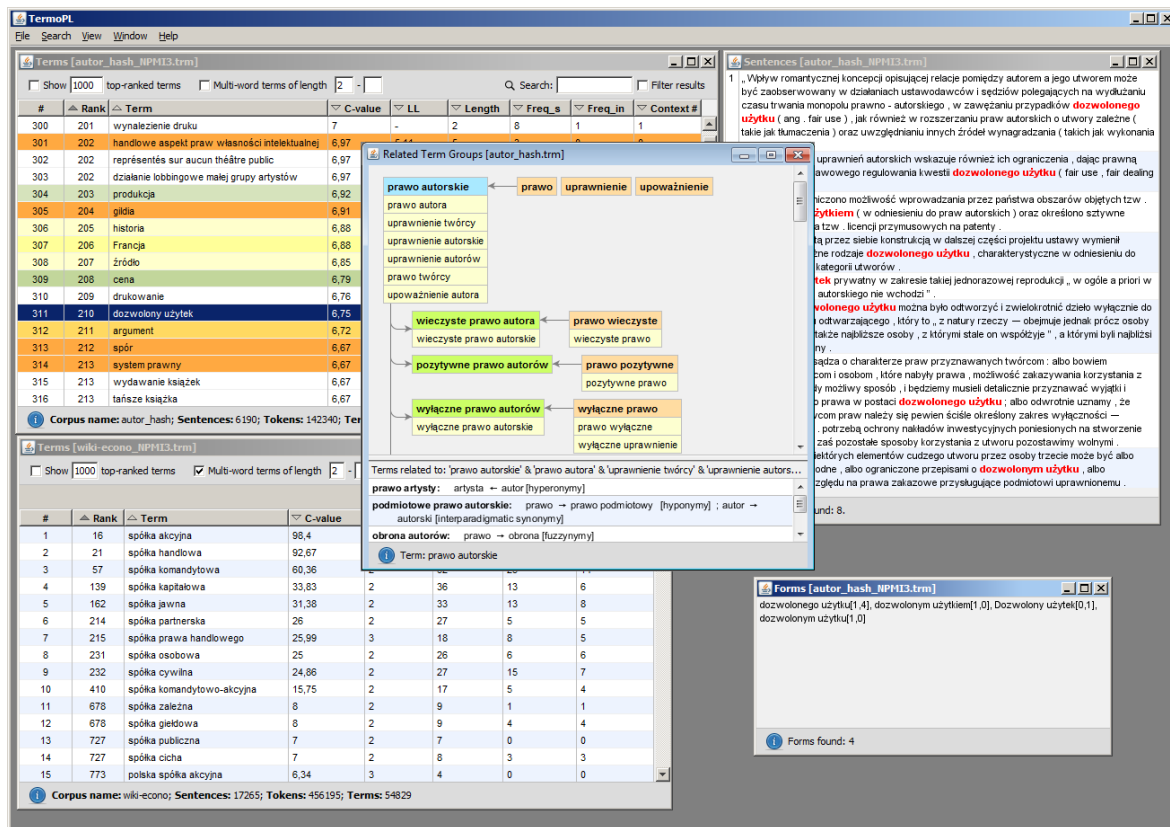


Figure 2: The virtual desktop for TernoPL.

occurrences (**Freq\_s**), number of occurrences within the context of another terms (**Freq\_in**) and the number of these contexts (**Context #**). The table can be sorted by any (except the first) column by clicking on its header. Columns may be sorted in ascending (▲) or descending (▼) order.

As it was mentioned before, the list of displayed terms can be truncated. The user may wish to view only multi-word terms or only those that are top-ranked. If the list of displayed terms is reduced to 1000 top-ranked terms, it might actually contain more entries as some of the terms may have assigned the same rank. The user may also limit the displayed list of terms by filtering those that contain a specified sequence of characters. Clicking the search icon (Q) reveals the available term filtering options (Figure 4). The user has three options to control the filtering process: he/she can choose a case-sensitive search, select whole-word strings, or define a java-style regular expression.

Truncation always starts from collecting all items that contain a specified sequence of characters, then all phrases with a length outside the specified range are removed, and finally the highest ranked terms are selected.

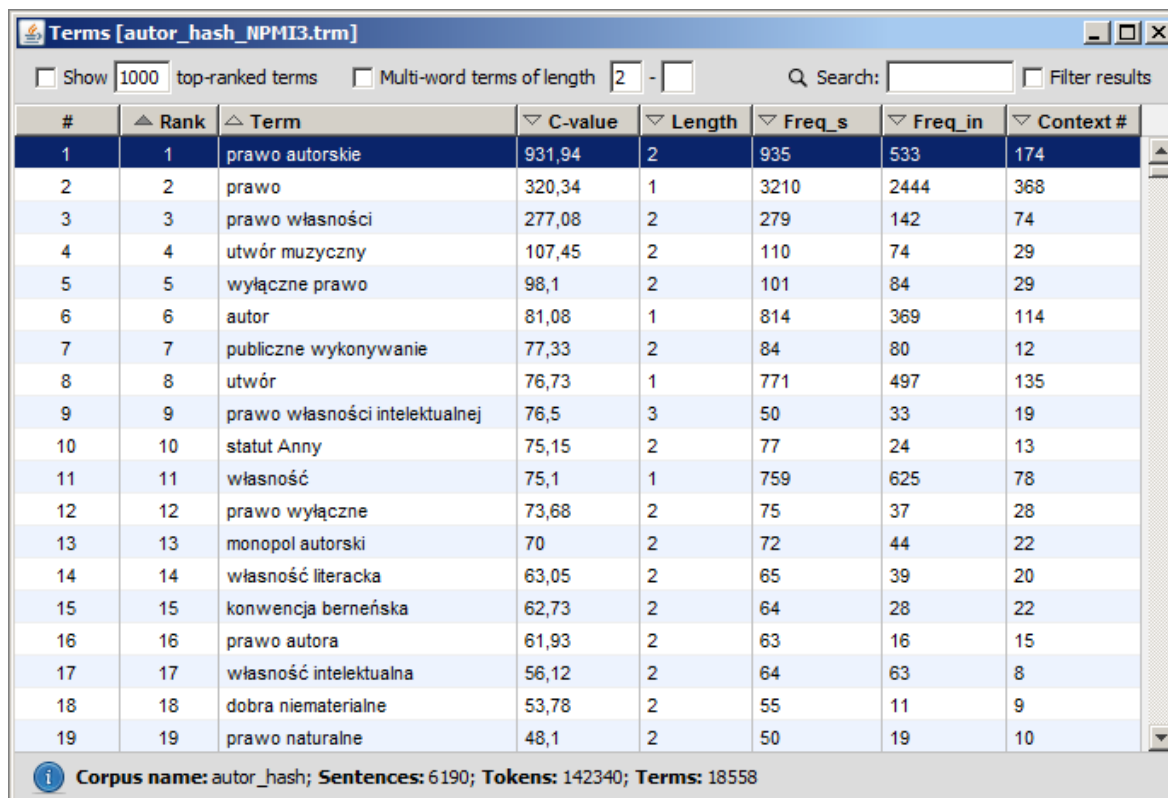
In case when the user decided to compare the extracted set of terms with other, previously extracted set of terms, the Terms window looks like on Figure 5.

In the window showing comparison results, the value of the chosen comparing coefficient is displayed just next to the right of the C-value. In Figure 5, log-likelihood values are shown.

The colours of the table's rows correspond to a term representativeness. All shades of yellow point out that a corresponding term is more representative for the domain corpus. Green colours show the opposite. The more saturated colour, the more representative a given term is for one of two corpora.

The Terms window is the main window for a document describing an extracted set of terms.

Closing the main window will automatically close all windows associated with one document, currently analysed set of terms.



#	Rank	Term	C-value	Length	Freq_s	Freq_in	Context #
1	1	prawo autorskie	931,94	2	935	533	174
2	2	prawo	320,34	1	3210	2444	368
3	3	prawo własności	277,08	2	279	142	74
4	4	utwór muzyczny	107,45	2	110	74	29
5	5	wyłączne prawo	98,1	2	101	84	29
6	6	autor	81,08	1	814	369	114
7	7	publiczne wykonywanie	77,33	2	84	80	12
8	8	utwór	76,73	1	771	497	135
9	9	prawo własności intelektualnej	76,5	3	50	33	19
10	10	statut Anny	75,15	2	77	24	13
11	11	własność	75,1	1	759	625	78
12	12	prawo wyłączne	73,68	2	75	37	28
13	13	monopol autorski	70	2	72	44	22
14	14	własność literacka	63,05	2	65	39	20
15	15	konwencja berneńska	62,73	2	64	28	22
16	16	prawo autora	61,93	2	63	16	15
17	17	własność intelektualna	56,12	2	64	63	8
18	18	dobra niematerialne	53,78	2	55	11	9
19	19	prawo naturalne	48,1	2	50	19	10

Corpus name: autor\_hash; Sentences: 6190; Tokens: 142340; Terms: 18558

Figure 3: The main window showing search results.

☐ Case sensitive
 ☒ Entire word
 ☐ Regular expression

Figure 4: Term filtering options.

Terms [autor\_hash\_NPMI3.trm]

Show 1000 top-ranked terms    Multi-word terms of length 2 -    Search:    Filter results

#	Rank	Term	C-value	LL	Length	Freq_s	Freq_in	Context #
300	201	wynalezienie druku	7	-	2	8	1	1
301	202	handlowy aspekt praw własności intelektualnej	6,97	5,11	5	3	0	0
302	202	représentés sur aucun théâtre public	6,97	-	5	3	0	0
303	202	działanie lobbingowe małej grupy artystów	6,97	-	5	3	0	0
304	203	produkcja	6,92	5,67	1	71	53	29
305	204	gildia	6,91	8,64	1	72	44	15
306	205	historia	6,88	0,72	1	71	54	25
307	206	Francja	6,88	3,04	1	70	7	6
308	207	źródło	6,85	0	1	71	53	21
309	208	cena	6,79	18	1	71	55	18
310	209	drukowanie	6,76	-	1	72	48	11
311	210	dozwolony użytek	6,75	-	2	8	5	4
312	211	argument	6,72	4,21	1	69	33	18
313	212	spór	6,67	5,08	1	69	35	15
314	213	system prawny	6,67	5,12	2	8	4	3
315	213	wydawanie książek	6,67	-	2	8	4	3
316	213	tańsza książka	6,67	-	2	8	4	3
317	214	ochrona prawa	6,5	1,71	2	8	6	4

Corpus name: autor\_hash; Sentences: 6190; Tokens: 142340; Terms: 18558; Compared with: wiki-econo\_NPMI3

Figure 5: Results of two corpora comparison.

$$-f_B(t)/f_A(t) \longleftrightarrow f_A(t)/f_B(t)$$

-9 <	[-9,-7)	[-7,-5)	[-5,-3)	[-3,-1)	[1,3)	[3,5)	[5,7)	[7,9)	>= 9
n/a					unique				

Figure 6: The meaning of table's colours.

## 6.2 The Forms Window

One may choose to collect all forms of extracted terms as they appear in an analysed corpus (see Section 6.8). In this case, selecting a row in the table of terms allows to display all forms of the corresponding term (see Section 6.9). For each form, the number of its occurrences as an independent and nested phrase is given in square brackets.

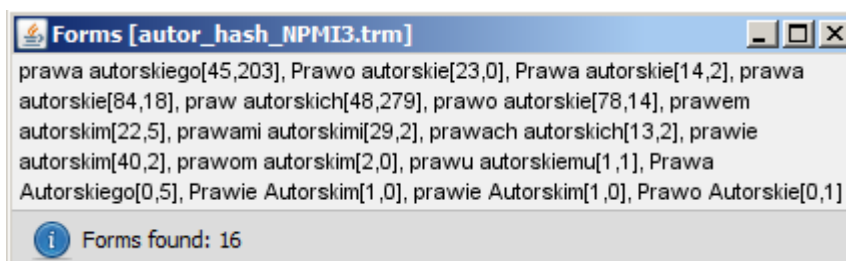


Figure 7: The window presenting all forms of a given term found in an analysed corpus.

## 6.3 The Sentences Window

If the user decided to index all sentences with extracted terms (see Section 6.8), selecting a row in the table allows to display all sentences in which the selected term appears (see Section 6.9).

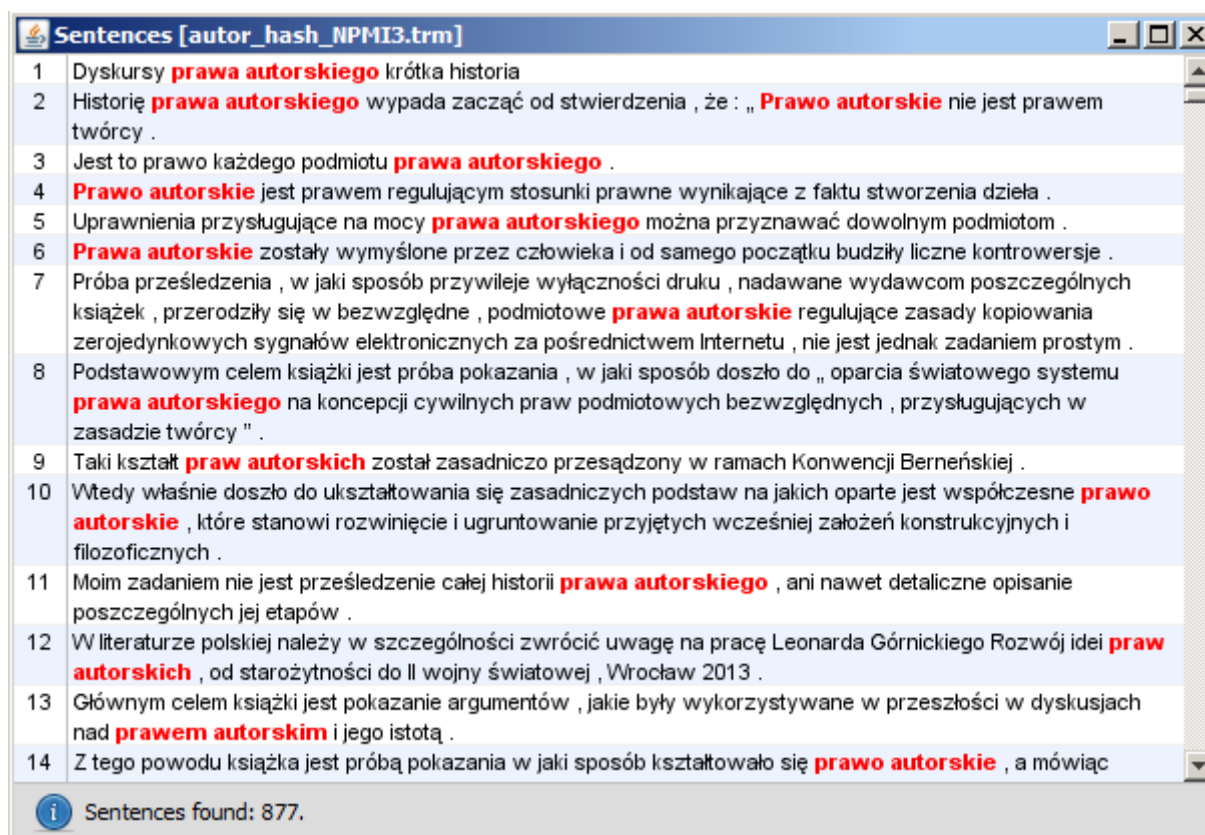


Figure 8: The window presenting all sentences containing a given term found in an analysed corpus.

## 6.4 The Related Term Groups Window

TermoPL has a functionality that enables semantic grouping of terms extracted from Polish texts [13]. The program combines each extracted term with less specific terms and identifies those that are semantically related to it using the information contained in plWordNet[3].

In the first stage of this process, for each extracted phrase, the program analyses phrases containing all of its words. A longer phrase may turn out to be a term that narrows down the meaning of the selected phrase, and a phrase of the same length may be synonymous with it. For example the phrases 'prawo autorskie' and 'autorskie prawo' (*copyright law*) are considered as synonymous, and the phrase 'amerykański rynek wydawniczy' (*the American publishing market*) is more specific than the phrase 'rynek wydawniczy' (*publishing market*). However, we cannot treat phrases as bags of words. For example, the term 'prawo autora' (*author's law*) contains the same words as the term 'autor prawa' (*author of the law*), but they do not mean the same thing. Grouping should link phrases built around a common head. For phrases 'prawo autora' and 'autor prawa', their heads are 'prawo' (*law*) and 'autor' (*author*), respectively. They are different, so these terms cannot be assigned to the same group.

The part of the phrase that should be converted to the nominative form is treated by TermoPL as the head of the phrase. As mentioned on page 3 in Section 2, the user can mark the head phrase with the ^ character when defining the grammar.

In the second stage of grouping terms, the program examines all possible combinations of replacing words in terms with words associated with them according to plWordNet. If the resulting phrase belongs to the set of generated terms, it is classified as a synonym or only as a term semantically related to the corresponding term. The category to which the term is assigned depends on the semantic relationship between the replaced words and the words that have replaced them. For grouping, the program uses several synset and lexical relations defined in plWordNet.

Table 4: Relation types used for grouping

category	relation ID	relation type
synonym	1	belonging to the same synset
	169	kind of interparadigmatic synonymy
related	10	hyponymy
	11	hyperonymy
	51	characterized by feature or state
	52	feature or state
	108	fuzzynymy of synsets
	148	similarity
	149	description
	242	role: material
	61, 62, 63, 141, 142, 244	kind of interparadigmatic synonymy

The window presenting groups of terms related to the selected term is split into two parts. The first part presents the hierarchy of terms related to this selected one. The lower a node is in the hierarchy, the more specific terms it represents. The second part lists those terms that are somehow related to the chosen one. This list can be changed by clicking any node from the hierarchy. When you double-click the term name in any node of the hierarchy or in the list of related terms, the term will be selected, and thus the content of the window will be updated.



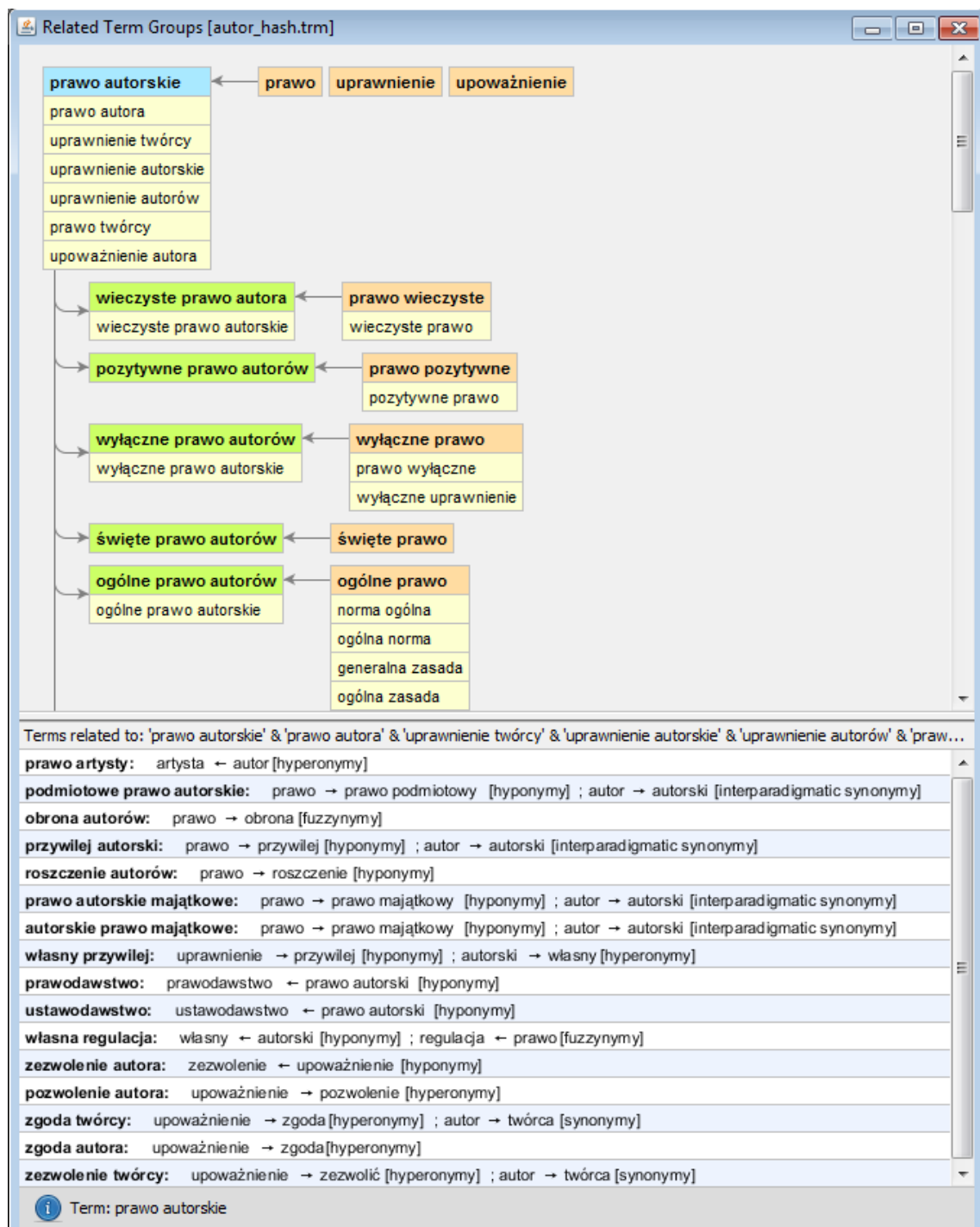


Figure 9: The window presenting groups of terms related to the term 'prawo autorskie'.

## 6.5 The Options Dialog

The user can change the behaviour of the program by setting different options. In the interactive mode, the initial values for the options are loaded from the file `.TermoPL-5`, which is created by the program in the user's home directory when it is run for the first time. This file is modified whenever the user changes some of the options and when the program terminates.

The program keeps several copies of options set. The first copy, let's call it the master copy, is created immediately when the program starts. This copy is then used whenever the user chooses to create a new set of terms. The copy of the master copy is associated with the newly created set of terms and will be used only with this set. Modifying the options for the currently used set of terms will modify only the options associated with this set and the master copy. Options associated with other sets of terms will remain unaltered.

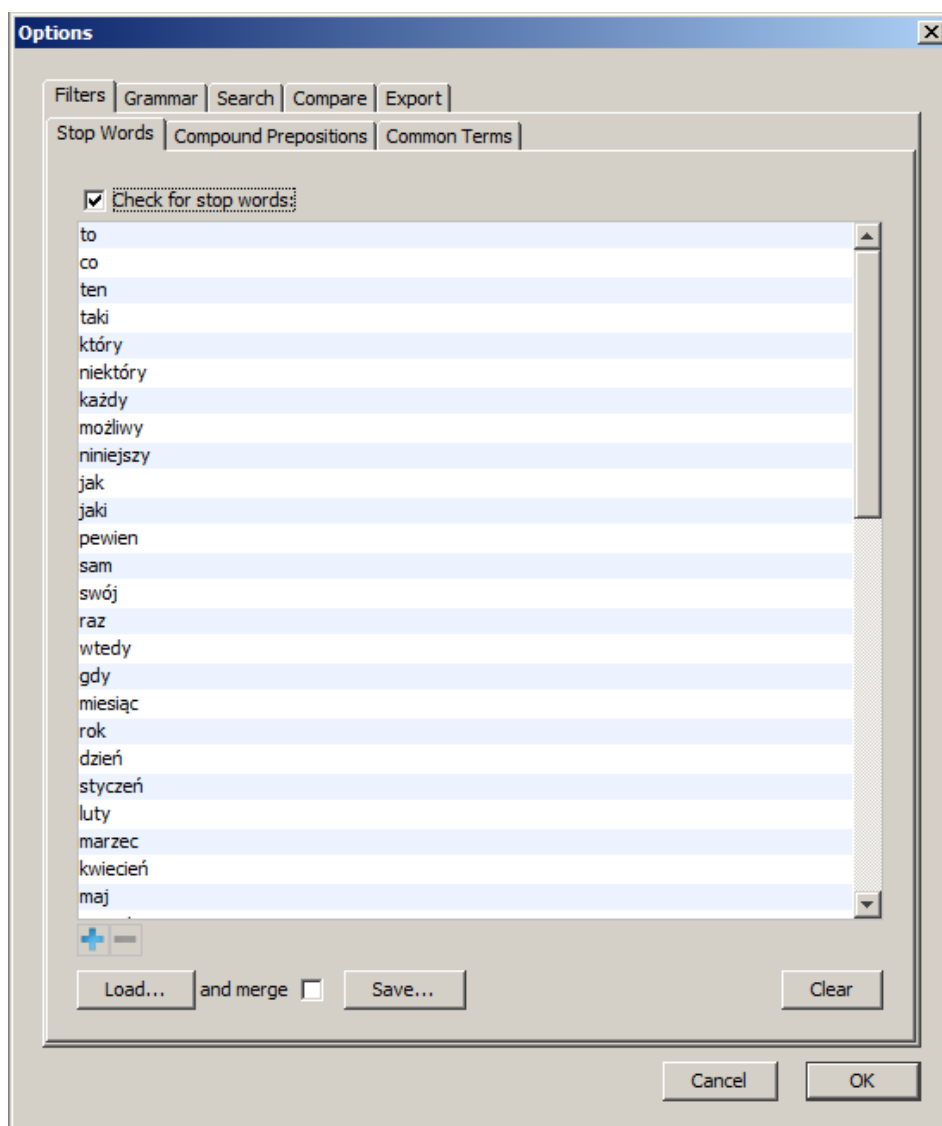


Figure 10: Optios – Filters – Stop Words.

The **Options** dialog consists of seven panels. In the first three panels we can define lists of filters: stop words (below), compound prepositions (page 18) and common terms (page 19).

These lists can be loaded [Load...] from UTF-8 encoded text files replacing existing lists, or loaded and merged (merge) with existing lists. Each list can be modified by the user. Double-clicking an item of a list calls a text editor. Clicking the buttons (+) and (-), adds a new entry or removes the selected item(s) from the list, respectively. Modified list can be saved [Save...] to a file.

The list of stop words consists of lemmatized forms. Each line of text in a file with stop words contains only one word. By default the list of stop words is empty. The default set of stop words can be loaded from `termopl_sw.txt`.

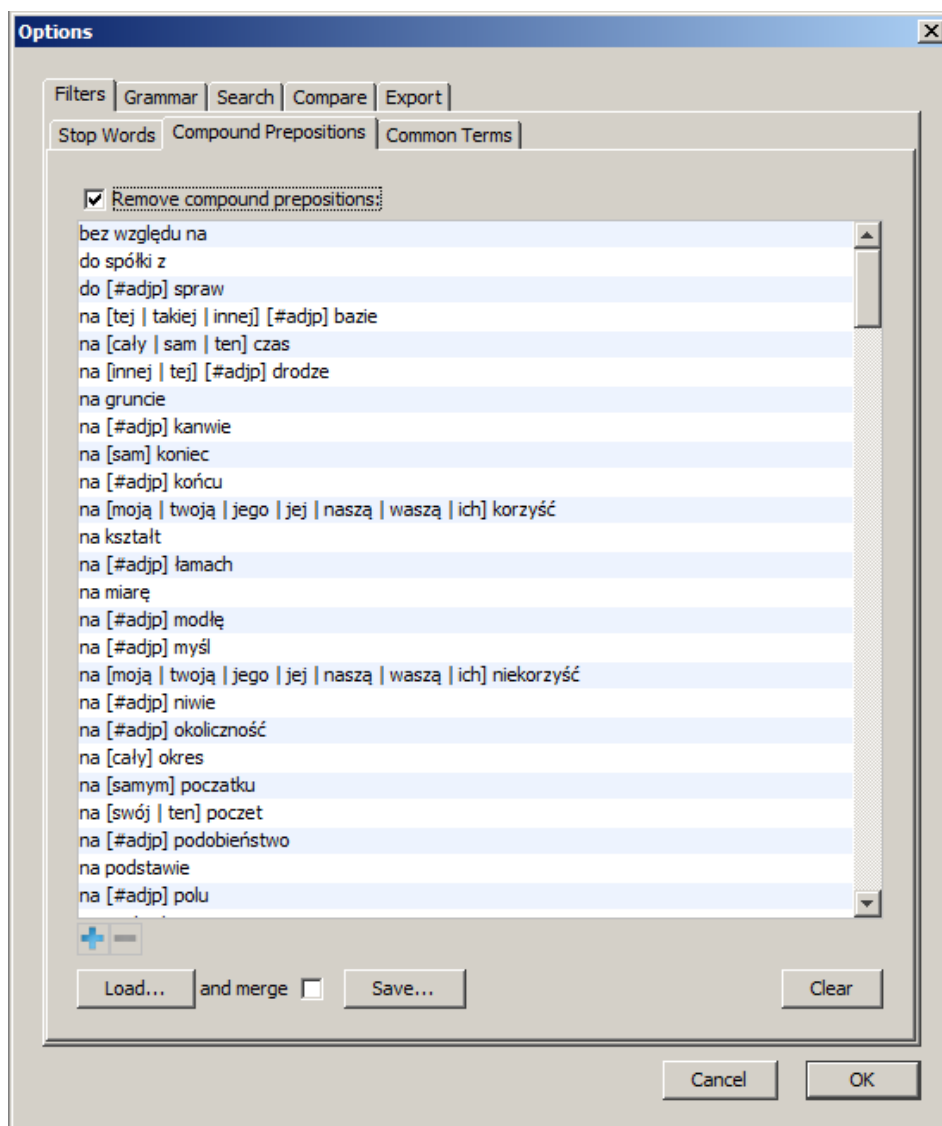


Figure 11: Options – Filters – Compound Prepositions.

The list of compound prepositions looks very much the same as the list of stop words. However, each line of a compound prepositions list defines a pattern. Each pattern contains obligatory and/or optional elements. For example, the pattern 'na [sam] koniec' has two obligatory elements 'na' and 'koniec', and only one optional element 'sam'. It will match expressions like 'na koniec' and 'na sam koniec'. Some of the elements define an alternative of words. For example, the pattern 'na [cały | sam | ten] czas' contains optional element being an alternative of three words: **cały**, **sam** and **ten**. It means that the whole pattern will

match expressions like 'na cały czas', 'na sam czas' and 'na ten czas'. If the user decides to make an alternative obligatory, it must put it in parenthesis '(...)' instead of brackets. The symbol #adjp frequently used in patterns matches a single adjective.

By default the list of compound prepositions is empty. The default set of compound prepositions can be loaded from `termopl_cp.txt`.

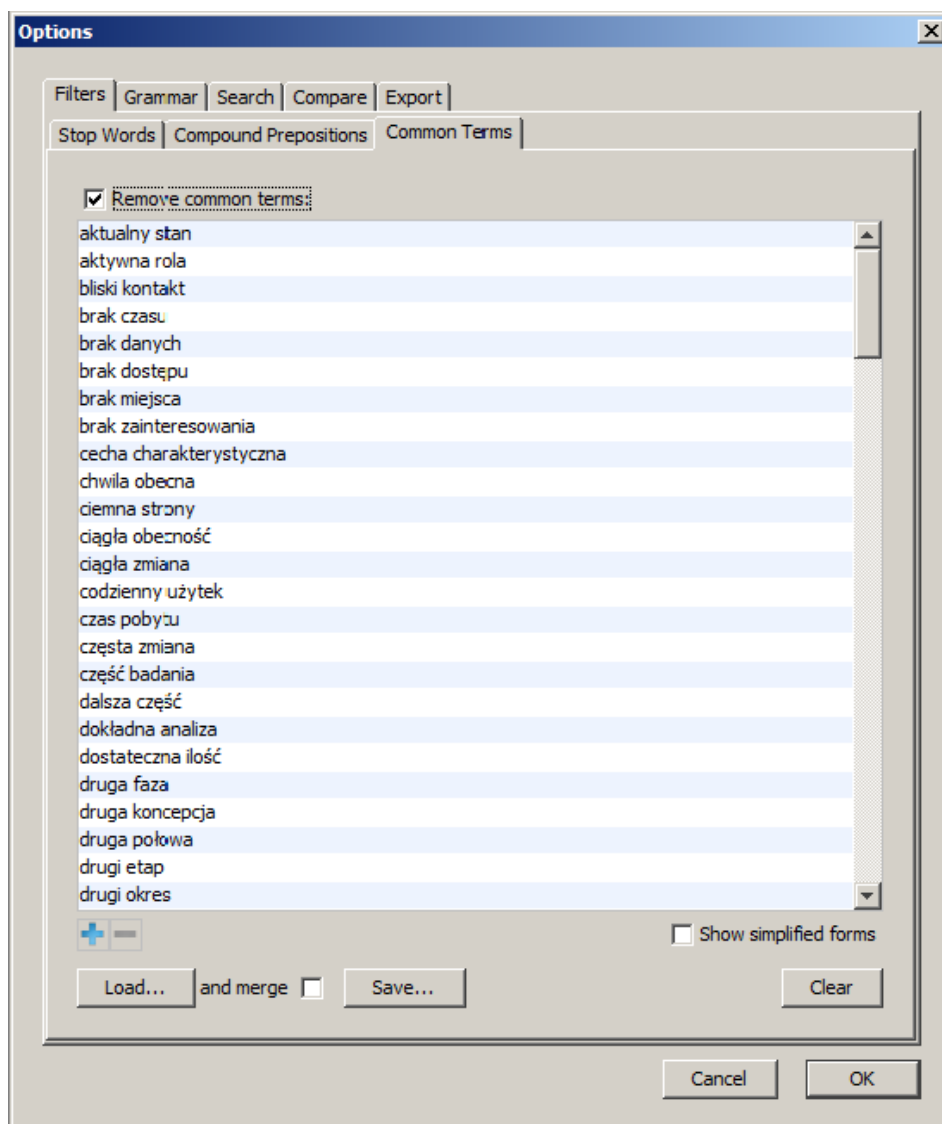


Figure 12: Optios – Filters – Common Terms.

The list of common terms is used to eliminate from the final list of extracted term candidates those which are general or out-of-domain. The user may choose to edit their simplified or base forms by selecting or deselecting the **Show simplified forms** check box. For new base forms, simplified forms are automatically generated. If we add a simplified form of a term, its base form becomes the same string as the simplified form.

Each line of a file with common terms contains a simplified form of a term and, optionally, its base form put in brackets.

By default the list of compound prepositions is empty. The default set of common terms can be loaded from `termopl_ct.txt`.

Working with base forms requires the Morfeusz 2.0 libraries to be installed.

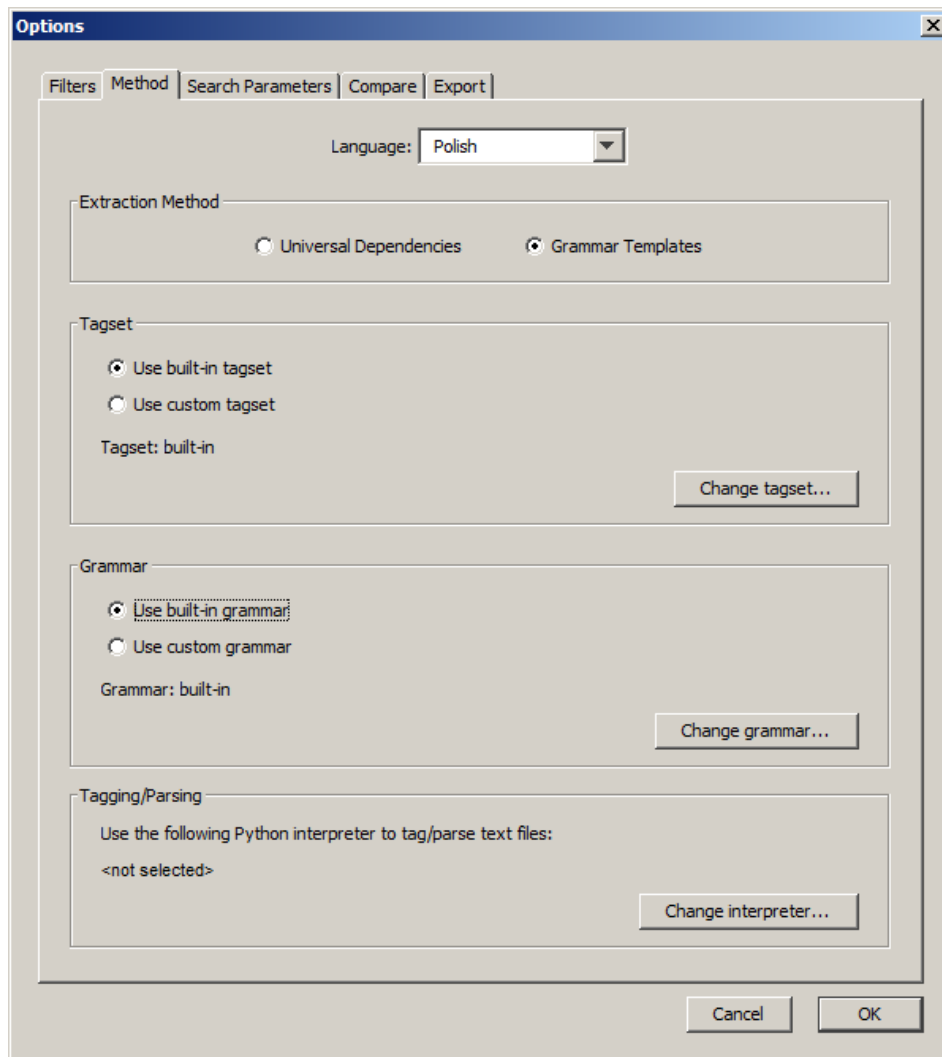


Figure 13: Options – Grammar.

Using this panel, we can inform the program about the language of the texts to be processed, the method used to extract terms, and decide which grammar and which tagset should be used.

The choice of language is important if the input is plain, untagged text. This information is passed to an external tool (the Stanza UD Parser), which will create a CoNLLu file(s) for further analysis. By choosing Polish, we enable a more precise method of converting terms into their base form. We can also take advantage of term grouping using plWordNet.

We can select alternative grammar by clicking on **[Change grammar...]** button and selecting one of the grammar files. By choosing the custom grammar we are allowed to use an alternative tagset. This tagset can be changed by clicking on **[Change tagset...]** button and selecting one of the tagset files. The use of Morfeusz to generate base forms of the extracted terms from Polish texts is possible only if the built-in tagset is selected. Grammar files as well as tagset files should be UTF-8 encoded text files.

To use the Stanza parser, it is necessary to indicate where the Python interpreter (see Section 8) with the stanza module installed is located. This can be done by clicking on **[Change interpreter...]** button and selecting the python executable.

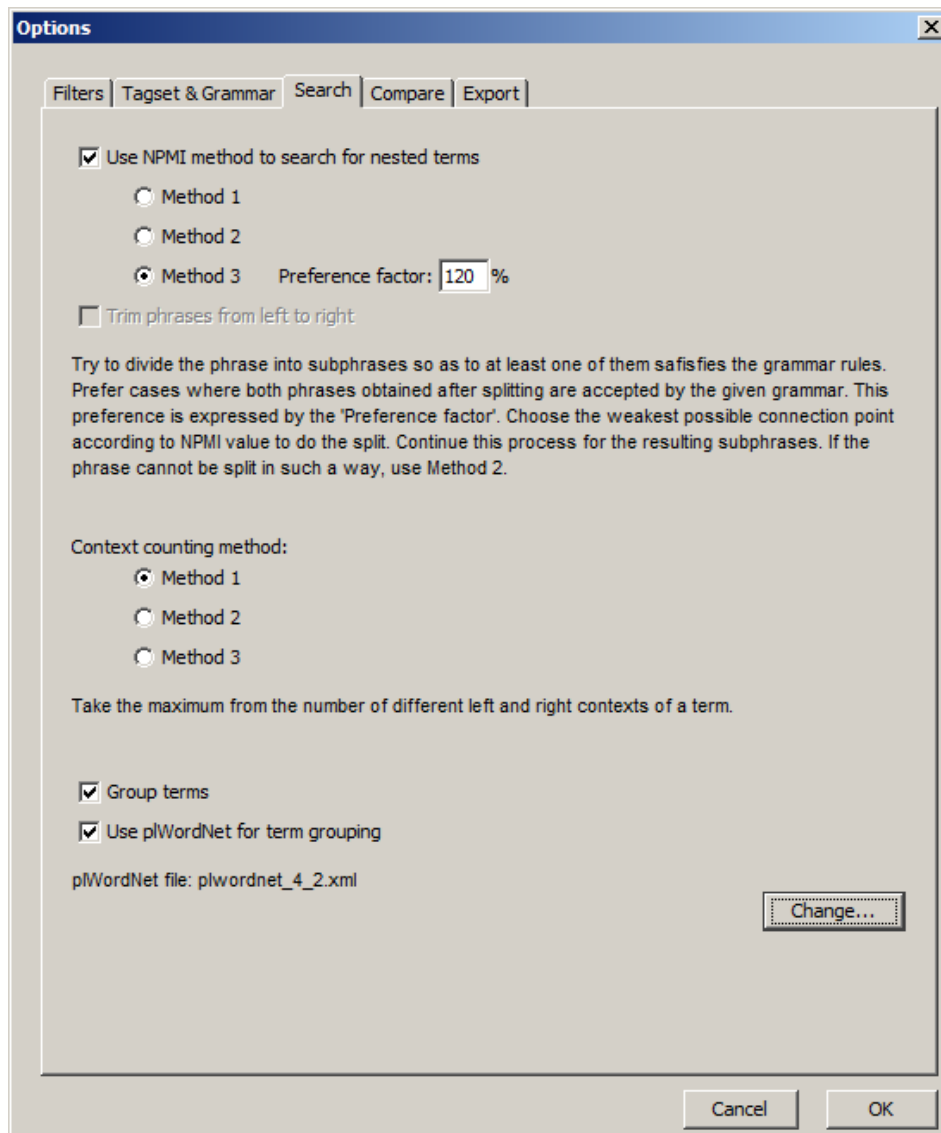


Figure 14: Options – Search.

The user should use this panel to set up the term extraction and context counting methods.

The user has three options when he/she chooses to use NPMI driven method to search for nested phrases. The first one always divides the phrase at the weakest connection point indicated by the lowest NPMI value and continues this process for the resulting subphrases even if they do not conform to the grammar rules. The second one is more sophisticated. It tries to divide the phrase into subphrases so as to at least one of them satisfies the grammar rules. It chooses the weakest possible connection point according to NPMI value to do the split and continue this process for the resulting subphrases. If the phrase cannot be split in such a way, the first method is used. The third method, which is the default method, also tries to divide the phrase into subphrases so as to at least one of them satisfies the grammar rules. However, it prefers the cases where both phrases obtained after splitting are accepted by the given grammar. This preference is expressed by the **Preference factor**. If the phrase cannot be split in such a way, the second method is applied. If the user decides not to use any of the NPMI methods, then all possible nested phrases are checked, unless the **Trim phrases from left to right** box is selected. In this case, nested phrases are generated by cutting out the initial fragment of a maximal phrase.

By a context of a nested phrase we will understand a pair  $(L, R)$ , where  $L$  and  $R$  are words located just before and right after the nested phrase. In some cases  $L$  or/and  $R$  can be empty. For example, any maximal phrase, which is not embedded in any larger acceptable phrase, has an empty context. The question is how we will count the contexts. The first method, which is the default method used by the program, counts different left and right contexts separately and then takes the maximum of these two values. The second method works similarly; however, it does not differentiate between pairs  $(L, R)$  and  $(R, L)$ . The last method counts all different pairs  $(L, R)$ .

Grouping of terms is performed when the **Group terms** check box is selected. To use plWordNet for grouping, the user must first download the appropriate xml file. This file is available at <http://plwordnet.pwr.wroc.pl/wordnet/download>.

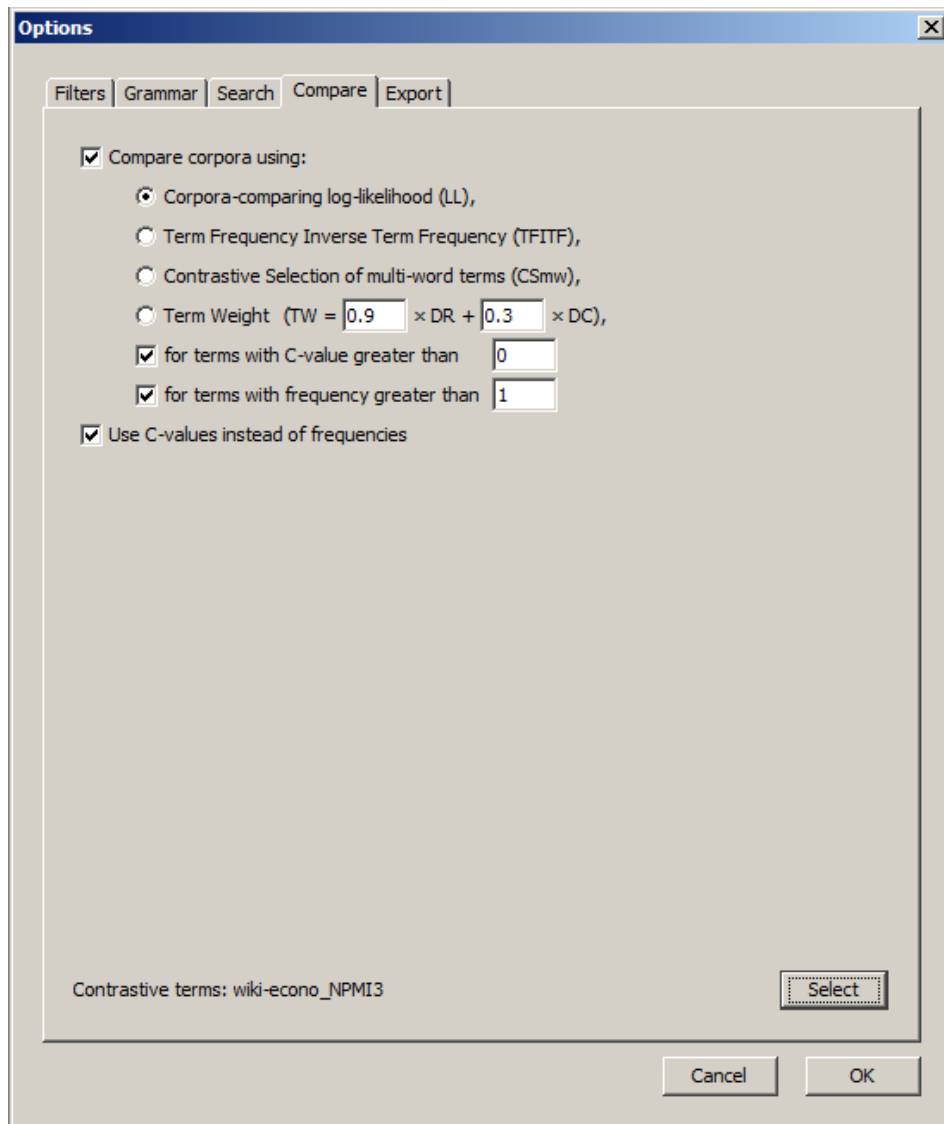


Figure 15: Options – Compare.

Using **Compare** panel we can set up corpora comparing methods and their parameters. They are described in Sections 3.3 – 3.6.

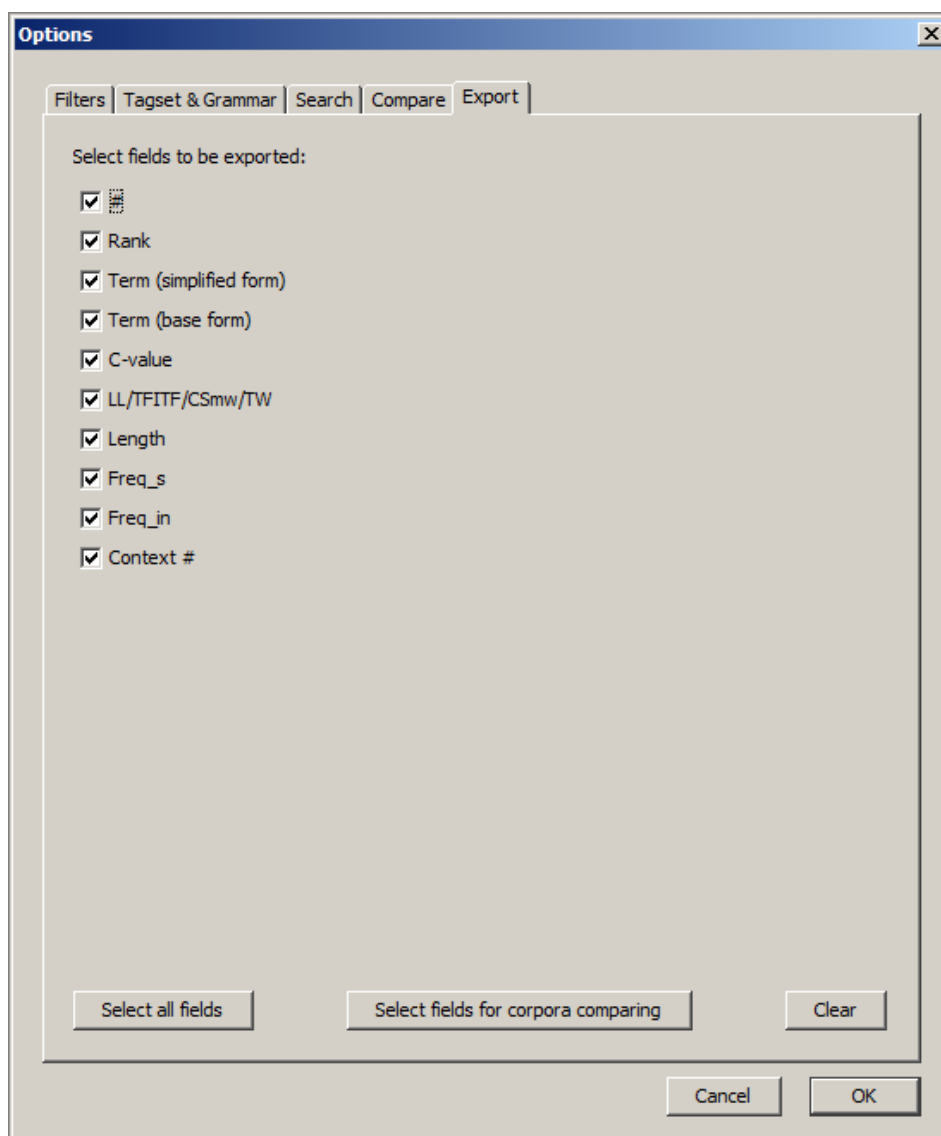


Figure 16: Options – Export.

The user can decide which columns from the results table will be exported to a text file. He/she can also decide whether to save all forms of terms collected by the program. Not every file format of saved terms is acceptable for corpora comparing. The simplest one contains exactly three fields: **Term (simplified form)**, **C-value** and **Freq\_s**. The other acceptable formats contain all fields from which we can exclude **Term (base form)** and/or **LL/TFITF/CSmw/TW**.

Figure 16 shows default settings for the **Export** panel.

## 6.6 Workspace

The location of all files used by Termopl for analysis is determined relative to the selected folder called the workspace. The user should place all analysed files in this folder. However, they can be arranged in subfolders. By default, the program working in interactive mode assumes that files are placed in TermoplWorkspace folder, which is automatically created in the user's home directory. It is possible to define multiple workspaces and they can be changed at any time. In batch mode, user has to define workspace explicitly. Otherwise, all input files



must be specified by their absolute paths.

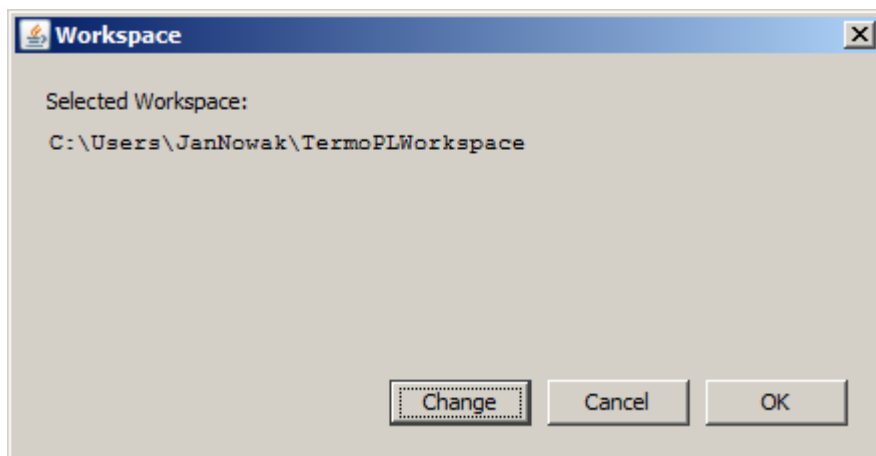


Figure 17: Workspace window

## 6.7 The File Menu

The **File** menu contains commands that allow to create, open, close, save, export and merge sets of terms.

To create a new set of terms one should choose the **New** command. The user will then be asked to select files for terms extraction.

With **Open** command we can choose existing set of terms to be loaded from a computer disk for further analysis. These files have extension `.trm`. By opening a `.trm` file we restore the list of extracted terms and all settings of the program, i.e. options and files used in the extraction process.

The **Merge...** command allows to combine the currently analysed set of terms with some other set from the disk. Merging two lists of terms is not equivalent to performing the entire extraction procedure on the combined set of files used to generate these lists. However, the results obtained with these two methods should not differ significantly. These differences result from how the NPMI is calculated (see Section 3.2). In the case of merging, NPMI is calculated separately for two sets of input files. When we perform the search on the combined set of input files, the NPMI is calculated for this combined set. The abovementioned methods are equivalent, if we resign from using the NPMI during extraction process. Merge removes all information about the term grouping.

The **Open Recent** submenu gives access to the recently opened set of terms.

To close the currently analysed set of terms we can use **Close** command. This is equivalent to closing the main window (see Section 6.1).

The **Save** and **Save As...** commands serve to save the list of terms together with the program's options and files used in the extraction process. Files saved with these commands can be further used for corpora comparing.

The **Workspace** command allows to change the current workspace.

The user can also export (the **Export...** command) the search results to UTF-8 encoded text file. The user can decide which fields of the terms table should be saved (see page 23). Only those terms are saved that are actually listed in the terms table. Exported search results can be subsequently used for corpora comparison.

The user may choose to save all forms of the extracted terms (the **Export Forms...** command) and sentences which contain them (the **Export Sentences...** command) to separate

UTF-8 encoded text files, provided that the appropriate information has been collected during the search process.

Identified term groups can be stored (the **Export Groups...** command) in an XML file of the following format:

```
<term-list>
  <term id = "t1" name = "xxx yyy zzz">
    <eq id = "e1" />
    ...
    <parent id = "p1" />
    ...
    <child id = "c1" />
    ...
    <rel id = "r1">
      <subst word = "w1" expr = "expr1" replaced = "s1"> (or <nosubst />)
        <reltype id = "n1" name = "xxx" />
        ...
      </subst>
      ...
    </rel>
    ...
  </term>
  ...
</term-list>
```

Realtion types are numbers from plWordNet.

On Microsoft Windows and Unix operating systems, the **File** menu contains the **Exit** command, which terminates the program. On Mac OS, the same function has the **Quit** command placed in the **TermoPL** menu located next to the **Apple** menu.

## 6.8 The Search Menu

The basic functions of this menu are term extraction (**Extract**) and comparing two sets of terms (**Compare**).

To perform a term extraction one has to load a corpus by selecting a file, a group of files, or a whole directory (the **Select File(s)...** command).

The **Select Contrastive Set of Terms** command serves to select data for comparison purposes. The same can be done with the **Options** dialog (see page 22).

Choosing the **Select File(s)** command displays a dialog (Figure 18) with which the user can create and edit the list of files used in the extraction of terms.

Clicking the buttons (+) and (-), adds a new entry or removes the selected item(s) from the list, respectively. Files that were deleted or moved to some other location are marked with (✖). Those located outside the current workspace are marked with (⚠).

The list is divided into two parts. The first part (highlighted in orange) contains files that were previously used to generate the currently analyzed set of terms. The second one contains new files that should be used in the subsequent search. After adding new files to the list, the user must decide whether to run the search only for new files or for all files in the list. The first case is equivalent to merging old set of terms with those extracted from new files (see the **Merge...** command described in Section 6.7). Removing any file from the old list causes that the extraction will be performed from scratch using all files from the list.

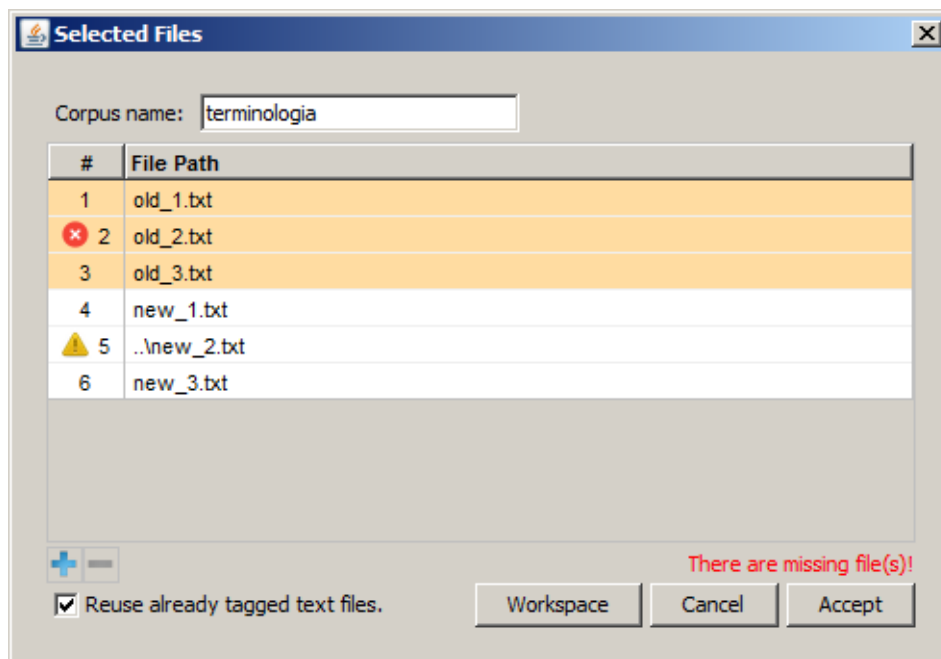


Figure 18: Selected Files

The **Corpus Name** field is filled up automatically but can be changed by the user at any time. For the newly created set of terms, this field becomes a part of the default file name for saving search results.

In case at least one of the selected files requires tagging, we can decide (**Reuse already tagged text files**) whether the program should save the tagged files so that they can be reused in the future. Saving tagged files speeds up the terms extraction process significantly. The tagged file is stored in the same directory as the source file under the source file name with the '.tgt' extension appended.

From the **Search** menu we can also set up various options used in the process of extraction. Choosing the **Preferences...** command will display the **Options** dialog, where the majority of search options can be selected. To conform to the look and feel of Mac OS, the **Preferences...** command has been moved to the **TermoPL** menu placed next to the **Apple** menu.

There are three options that can be altered directly from the **Search** menu. We can instruct the program to calculate base forms of the extracted terms (the **Calculate Base Forms** command). This menu item is available only when the built-in tagset is in use (see page 20).

The user may wish to collect all forms of extracted terms (the **Collect All Analysed Forms of Terms** command) and/or all sentences containing them (the **Index Sentences with Extracted Terms** command).

## 6.9 The View Menu

With this menu we can access auxiliary information about a selected term in the main window. The user may wish to check forms of the selected term as they appear in the analysed corpus (the **Forms of Selected Term** command), view sentences from the corpus containing that term (the **Sentences with Selected Term** command), or view groups of terms associated with the selected term (the **Related Term Groups** command). As it is explained in Section 6.8, To access this information it is necessary to instruct the program to collect it.

The **View** menu also allows to change the size of the application font (the **Increase Font** and **Decrease Font** commands).

## 6.10 The Window Menu

While working with Termopl, many windows may be open at one time. The Window menu helps tidying up window clutter that can easily occur when several sets of terms are processed simultaneously.

With **Show One Document** command we can display only windows for the currently analysed set of terms. In this case, all windows associated with other documents are hidden. The **Show All Documents** command will display all windows of all open documents again.

The **Document Switch List** contains a list of opened term sets and allows you to select the one you want to view. In case when only one document is allowed to display its contents, selecting a document from this list shows all its open windows and hides the currently analysed document. Otherwise, all windows of the selected document are placed in front of all other windows.

The **Cascade** command arranges all opened Termopl windows one over another such that for all covered windows only the title bars are visible.

The **Tile** command organises all open Termopl windows so that they do not overlap, if possible. It is most useful when only a few windows are opened and each of them takes up a reasonable screen area.

The rest of the menu items refer to open windows of the currently analysed set of terms. Selecting such an item displays the corresponding window in front of other windows.

## 6.11 The Help Menu

This provides access to general information about the Termopl software (the **About Termopl** command) as well as to the user manual (the **User Manual** command). On Mac OS X, the command **About Termopl** is moved to the **Termopl** menu located next to the **Apple** menu.

## 7 Batch processing

Termopl can be run in a batch mode. To invoke the program in a batch mode the user should enter the following command:

```
> java [JVM options] -jar Termopl.jar [program options] file...
```

submitting at least one file to be processed or specifying any of the program's options.

For Unix, use the following command:

```
> java -Djava.library.path=/usr/lib/jni/ -jar Termopl.jar [program options] file...
```

For some reasons, invoking the program under Windows sometimes requires the `-Djava.library.path` option.

```
> java -Djava.library.path=. -jar Termopl.jar [program options] file...
```

We can submit two types of files for batch processing: those that will be searched for new terms, and those that contain already extracted lists of terms (`.trm` files). Termopl will first merge lists of terms from `.trm` files and then modify the resulting list with new terms extracted from remaining input files. If any `.trm` file is specified as input, all options listed in the argument list and `.conf` file will be ignored. In this case the program will take settings saved in the first `.trm` file listed in the argument list.

A configuration file may contain any of the above options except `-conf`. Options declared in a command line supersede those defined in a configuration file. If no configuration file is specified, the program checks whether the default configuration file `termopl_conf.txt` is available in a directory where Termopl is installed.

option	argument(s) and meaning
-conf	configuration file (this option cannot be used in a configuration file)
-wrk	workspace
-wdn	WordNet .xml file
-in	input file(s) (this option cannot be used in a command line)
-out	binary file with extracted terms
-exp	UTF-8 encoded text file with extracted terms
-expf	UTF-8 encoded text file with all forms of extracted terms; option -nf will be ignored
-expg	XML file
-exps	UTF-8 encoded text file with sentences containing extracted terms; option -indx will be switched on
-comp	file with contrastive set of terms
-sw	file with stop words
-SW	use default set of stop words (termopl_sw.txt)
-cp	file containing compound prepositions
-CP	use default set of compound prepositions (termopl_cp.txt)
-ct	file with common terms
-CT	use default set of common terms (termopl_ct.txt)
-grammar	file with a user-defined grammar
-tagset	file with a user-defined tagset
-python	path to the Python interpreter with the stanza package installed
-lang	ISO 639-1 Code (default is pl)
-method	ud – Universal Dependencies, gt – Grammar Templates
-det	include, exclude, article
-detr	(default is 20)
-mw	save multi-word terms only
-tr	the number of top-ranked terms to be saved
-sf	use simplified forms
-nf	do not collect all forms of terms
-indx	index sentences with extracted terms
-sort	sort table of results by a selected column in ascending order; select one of the following columns: rank, term, cvalue, comp, length, freq_s, freq_in, or context
-SORT	sort table of results by a selected column in descending order; select one of the following columns: rank, term, cvalue, comp, length, freq_s, freq_in, or context
-srch	NPMI methods: npmi1, npmi2 or npmi3; nonpmi1 – find all term candidates; nonpmi2 – find term candidates by trimming phrases from left to right;
-cntx	context counting method; 1, 2 or 3 – use one of the context counting methods
-cc	corpora comparing; no – do not perform any comparisons, or use one of the following methods: ll, tfitf, csmv or tw
-pf	the number defining the preference factor used by the third NPMI method
-frq	use frequencies instead of C-values while comparing corpora
-freq	consider terms with a frequency greater than the given number while comparing corpora
-cval	consider terms with a C-value greater than the given number while comparing corpora

*continued...*

option	argument(s) and meaning
-save	fields to export: #, rank, sf (term's simplified form), bf (term's base form), cvalue, comp, length, freq_s, freq_in, context

We can submit two types of files for batch processing: those that will be searched for new terms, and those that contain already extracted lists of terms (.trm files). TermopL will first merge lists of terms from .trm files and then modify the resulting list with new terms extracted from remaining input files. If any .trm file is specified as input, all options listed in the argument list and .conf file will be ignored. In this case the program will take settings saved in the first .trm file listed in the argument list.

A configuration file may contain any of the above options except -conf. Options declared in a command line supersede those defined in a configuration file. If no configuration file is specified, the program checks whether the default configuration file `termopl_conf.txt` is available in a directory where TermopL is installed.

Using the options -SW, -CP and -CT requires appropriate default sets of filters to be placed in a directory where the program TermopL itself is installed.

Invoking the program without any program option and an empty file list:

```
> java [JVM options] -jar TermopL.jar
```

causes the program to run in the interactive mode.

## 8 Requirements

TermopL is written in Java programming language and therefore requires Java Runtime Environment<sup>5</sup>(version 8 or later) to be installed on a target machine (Windows, Linux or Mac OS X). Since TermopL uses Morfeusz 2.0<sup>6</sup> to generate base forms of terms and produce simplified forms for the list of common terms, all libraries of Morfeusz 2.0 have to be installed, too. As long as the user does not need to work on base forms, Morfeusz 2.0 libraries are not required.

To enable the tagging/parsing of plain text files one has to install the Python interpreter<sup>7</sup>. A script written in Python that creates CoNLLu files was tested using Python version 3.11.5. After installing Python, install the stanza module by running the following command:

```
> pip install stanza.
```

The program is distributed as an executable jar file, so it can be started by double-clicking on its icon.

The program requires about 1GB of RAM to process corpora of approximate size of 500 000 tokens. For considerably bigger data, one should reserve more memory invoking the program with -Xmx and -Xms Java options, e.g.:

```
> java -Xmx5G -Xms4G -jar TermopL.jar,
```

which reserves minimum 4GB and up to 5GB of memory for the program to run.

The Python script that converts the files to CoNLLu format requires an additional approximately 1 GB of RAM.

---

<sup>5</sup>Java Runtime Environment can be downloaded from <https://www.java.com/en/download/>

<sup>6</sup>Morfeusz 2.0 is available on <http://sgjp.pl/morfeusz/dopobrania.html>

<sup>7</sup>Python interpreter can be downloaded from <https://www.python.org/downloads/>

## References

- [1] Francesca Bonin, Felice Dell’Orletta, Simonetta Montemagni, and Giulia Venturi. A contrastive approach to multi-word extraction from domain-specific corpora. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*. European Language Resources Association, 2010.
- [2] Gerlof Bouma. Normalized (Pointwise) Mutual Information in Collocation Extraction. In *Proceedings of the Biennial GSCL Conference 2009*, pages 31–40, Tübingen, 2009. Gesellschaft für Sprachtechnologie & Computerlinguistik.
- [3] A. Dziob, M. Piasecki, and E. Rudnicka. plwordnet 4.1—a linguistically motivated, corpus-based bilingual resource. In C. Fellbaum, P. Vossen, E. Rudnicka, M. Maziarz, and M. Piasecki, editors, *Proceedings of the 10th Global WordNet Conference: July 23-27, 2019, Wrocław (Poland)*, pages 353–362. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2019.
- [4] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. Automatic Recognition of Multi-Word Terms: the C-value/NC-value Method. *Int. Journal on Digital Libraries*, 3:115–130, 2000.
- [5] Nancy Ide, Patrice Bonhomme, and Laurent Romary. XCES: An XML-based encoding standard for linguistic corpora. In M. Gavrilidou, G. Carayannis, S. Markantonatou, S. Piperidis, and G. Stainhauer, editors, *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC’00)*, Athens, Greece, May 2000. European Language Resources Association (ELRA).
- [6] Małgorzata Marciniak and Agnieszka Mykowiecka. NPMI driven recognition of nested terms. In *Proceedings of the 4th International Workshop on Computational Terminology (Computerm)*, pages 33–41. Association for Computational Linguistics and Dublin City University, 2014.
- [7] Małgorzata Marciniak, Agnieszka Mykowiecka, and Piotr Rychlik. Termopl - a flexible tool for terminology extraction. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, may 2016. European Language Resources Association (ELRA).
- [8] Małgorzata Marciniak, Piotr Rychlik, and Agnieszka Mykowiecka. TermoUD – a language-independent terminology extraction tool. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 178–186, Dubrovnik, Croatia, 2023. Association for Computational Linguistics.
- [9] Roberto Navigli and Paola Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics*, 30(2):151–179, 2004.
- [10] Adam Przepiórkowski, Mirosław Bańko, R. L. Górski, and Barbara Lewandowska-Tomaszczyk, editors. *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warszawa, 2012.

- [11] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.
- [12] Paul Rayson and Roger Garside. Comparing corpora using frequency profiling. In *Proceedings of the Workshop on Comparing Corpora - Volume 9, WCC '00*, pages 1—6, 2000.
- [13] Piotr Rychlik, Małgorzata Marciniak, and Agnieszka Mykowiecka. TermoPL: A tool for extracting and clustering domain related terms. In *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*, pages 1–4, New York, NY, USA, 2022. Association for Computing Machinery.
- [14] Marcin Woliński. Morfeusz reloaded. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pages 1106–1111, Reykjavík, Iceland, 2014. ELRA.