

TermoPL* 5.6

user manual

Institute of Computer Science
Polish Academy of Sciences

23.08.2021

*TermoPL is co-funded by CLARIN-PL

1 Introduction

TermoPL is a tool created to extract terminology from domain corpora in Polish. The program extracts noun phrases, term candidates, with the help of a simple grammar that can be adapted for user's needs. It applies the C-value method to rank term candidates being either the longest identified nominal 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. We support the recognition of nested terms by word connection strength which allows us to eliminate truncated phrases from the top part of the term list. The program has an option to convert simplified forms of phrases into correct phrases in the nominal 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 also compare two candidate term lists using four different coefficients showing asymmetry of term occurrences in this data.

You can learn more about TermoPL from [5].

2 How it works?

TermoPL searches a given set of texts and creates a list of forms that might be considered as candidates for terms characteristic for a chosen domain. The program accepts an UTF8 encoded input with morphosyntactic analysis in various formats such as NKJP [7], XCES, 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.

Starting from version 5.1, the program also accepts files of CoNLL-U format¹, which is frequently used by several dependency parsers.

For Polish it is also possible to use untagged input files. In this case, the files are first processed by the Concraft [9] tagger.

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

¹<https://universaldependencies.org/format.html>

$$\begin{aligned}
NPP &: \$NAP \ NAP_GEN^*; \\
NAP[agreement] &: AP^* \ N \ AP^*; \\
NAP_GEN[case = gen] &: NAP; \\
AP &: ADJ \mid ADJA \ DASH \ ADJ \mid PPAS; \\
N[pos = subst, ger]; \\
ADJ[pos = adj]; \\
ADJA[pos = adja]; \\
PPAS[pos = ppas]; \\
DASH[form = "-"];
\end{aligned}$$

Figure 1: The built-in grammar.

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 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 (see Section 3.2) driven recognition of nested phrases introduced in [4]. 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 same factor. By default, TermoPL sets this factor to 120%.

All sequences recognised in this way are converted into simplified forms, in which all words are lemmatized and stored in a set representing term candidates. 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 romańskiej_{loc,sg}*, *katedrom romańskim_{dat,pl}*) depending on the case and number. Two of these forms are homomorphic with the other ones.

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.

Two lists are associated with each element of the set — the optional one containing all different orthographic forms of the term, and the other containing all distinct contexts in which these forms appear. The second list is automatically deleted after C-values are calculated. The first list, although it is optional, may play an important role when the base forms of terms are generated. This functionality, however, can be used only if the standard tagset is selected (see

4 and 12).

Additionally, for each term, two values are computed: the total number of term occurrences in the corpus and the number of occurrences within other, longer terms. Having all this information, the program calculates the C-value for each term and sorts the list of term candidates from the highest to the lowest C-value. Finally, if the user wishes to do so, simplified forms are replaced by base forms of the terms.

To obtain base forms a token or a group of tokens matched with a symbol marked with the \$ character are replaced by their nominal 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 nominal forms, whereas *NAP_GEN* phrases are left as they appear.

In this process, the new version of Morfeusz [10, the morphosyntactic analyser and generator for Polish] is used. 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. In a case where the user decided not to collect all orthographic forms, the process of converting simplified forms to base forms relies solely on Morfeusz.

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 results of term extraction can be saved to a file, which in turn may be used to make comparisons with other corpora. The program calculates a selected measure for corpora similarity (see Sections 3.3–3.6) and marks out listed terms with different shades of colours according to their representativeness in analysed corpora.

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

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 [3]. 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 6.4 (page 14).

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 [8] 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

Contractive 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) [6] 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 α and β 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 α and β 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 Project², 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:

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

²https://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 2.0³. 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_i**) 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 : cat₀ : ... : cat_{i-1} : : cat_{i+1} : ... : cat_n.

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.

³<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
fin	forma nieprzeszła	number:person:aspect
bedzie	forma przyszła czasownika BYĆ	number:person:aspect
aglt	aglutynant czasownika BYĆ	number:person:aspect:vocality
praet	pseudomiesłów	number:person:gender:aspect:aggl
impt	rozkaznik	number:person:aspect
imps	bezosobnik	aspect
inf	bezokolicznik	aspect
pcon	imiesłów przysłówkowy współczesny	aspect
pant	imiesłów przysłówkowy uprzedni	aspect
ger	odśownik	number:case:gender:aspect:negation
pact	imiesłów przymiotnikowy czynny	number:case:gender:aspect:negation
ppas	imiesłów przymiotnikowy bierny	number:case:gender:aspect:negation
winien	czasownik typu WINIEN	number:gender:aspect
pred	predykatyw	—
subst	rzeczownik	number:case:gender:sgender
depr	rzeczownik – forma deprecjatywna	number:case:gender
adj	przymiotnik	number:case:gender:degree
adja	przymiotnik przyprzymiotnikowy	—
adjp	forma poprzyimkowa	case
adjc	przymiotnik predykatywny	—
adv	przysłówek	degree
num	liczebnik	number:case:gender:accom:sgender
ppron12	zaimek nietrzecioosobowy	number:case:gender:person:accent

ppron3	zaimek trzecioosobowy	number:case:gender:person:accent:pprep
siebie	zaimek SIEBIE	case
prep	przyimek	case:vocality
conj	spójnik współrzędny	—
comp	spójnik podrzędny	—
brev	skrót	fstop
interj	wykrzyknik	—
part	partykuła	vocality
frag	człon frazeologizmu	—
interp	interpunkcja	—
ign	forma nierozpoznana	—

Table 2: Grammatical classes in the default tagset

category	meaning (in Polish)	values
number	liczba	sg, pl
case	przypadek	nom, gen, dat, acc, inst, loc, voc
gender	rodzaj	m1, m2, m3, f, n
sgender	przyrodzaj	pt, ncol, col
person	osoba	pri, sec, ter
degree	stopień	pos, com, sup
aspect	aspekt	imperf, perf
negation	zanegowanie	aff, neg
accent	akcentowość	akc, nakc
pprep	poprzyimkowość	praep, npraep
accom	akomodacyjność	congr, rec
aggl	aglutynacyjność	nagl, agl
vocality	wokaliczność	wok, nwok
fstop	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 2, `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 7). 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 since version 4. The user may open and analyse several set 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 3. 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 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

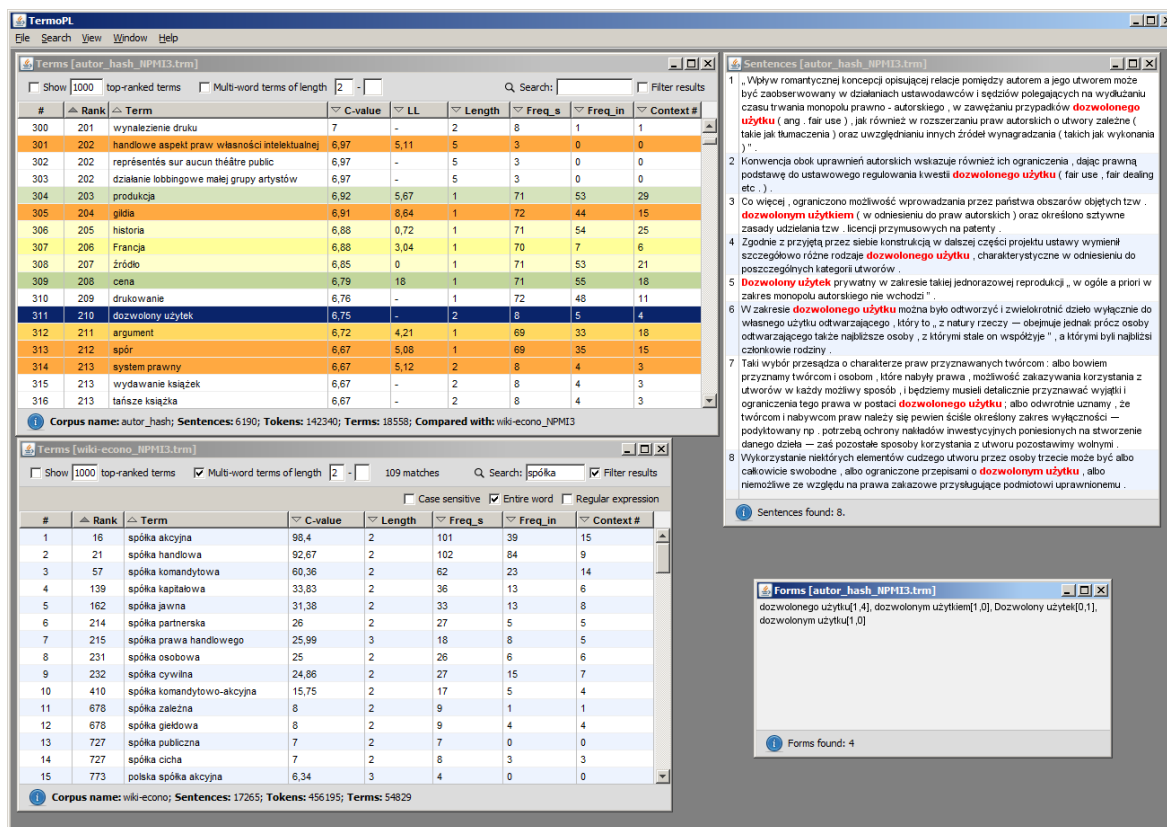


Figure 2: The virtual desktop for TernoPL.

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.

Terms [autor_hash_NPMI3.trm]

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

#	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	dobro 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]								
<input type="checkbox"/> Show 1000 top-ranked terms		<input type="checkbox"/> Multi-word terms of length 2 -		Q Search:		<input type="checkbox"/> 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 6.7). In this case, selecting a row in the table of terms allows to display all forms of the corresponding term (see Section 6.8). For each form, the number of its occurrences as an independent and nested phrase is given in square brackets.

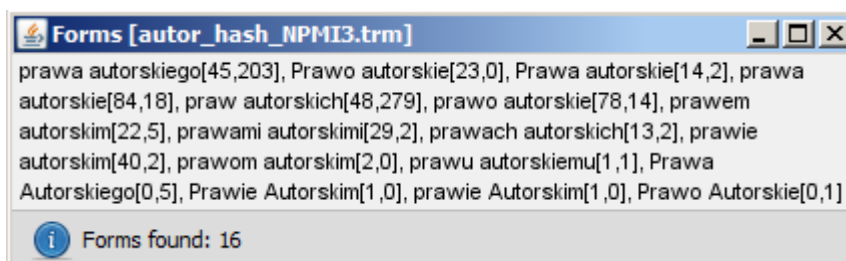


Figure 7: The window presenting all forms of a given term found in an analysed corpus.

6.3 The Sentences Window

If the user decided to index all sentences with extracted terms (see Section 6.7), selecting a row in the table allows to display all sentences in which the selected term appears (see Section 6.8).

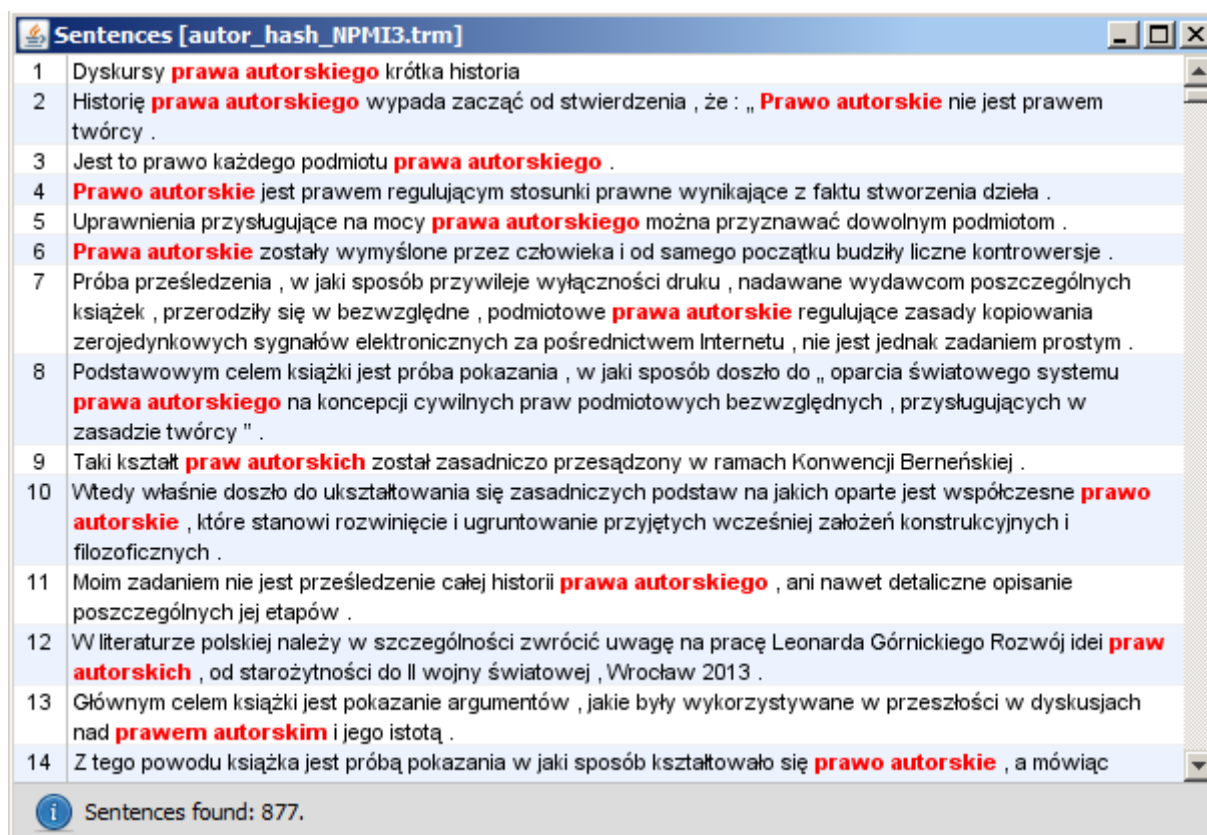


Figure 8: The window presenting all sentences containing a given term found in an analysed corpus.

6.4 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.

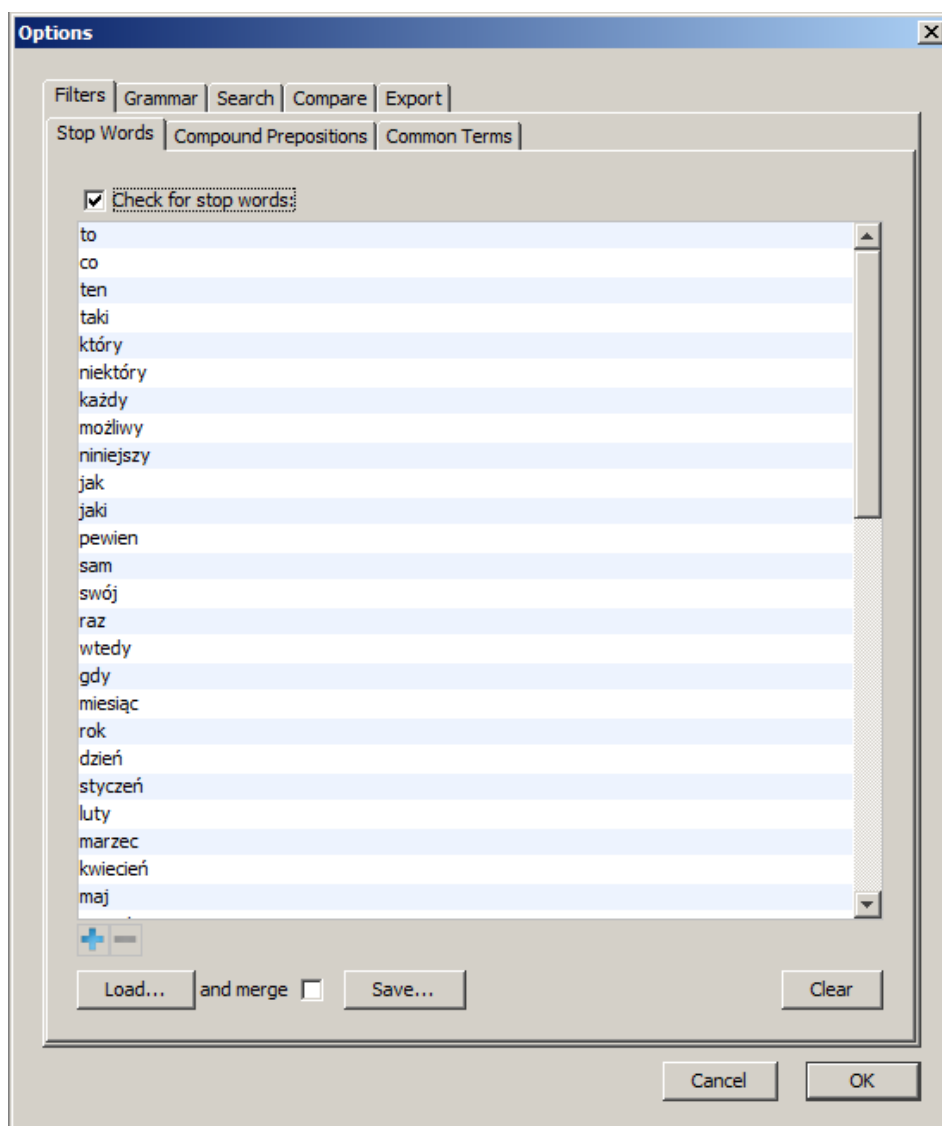


Figure 9: Optios – Filters – Stop Words.

The **Options** dialog consists of seven panels. In the first three panels we can define lists of filters: stop words (below), compound prepositions (page 15) and common terms (page 16).

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`.

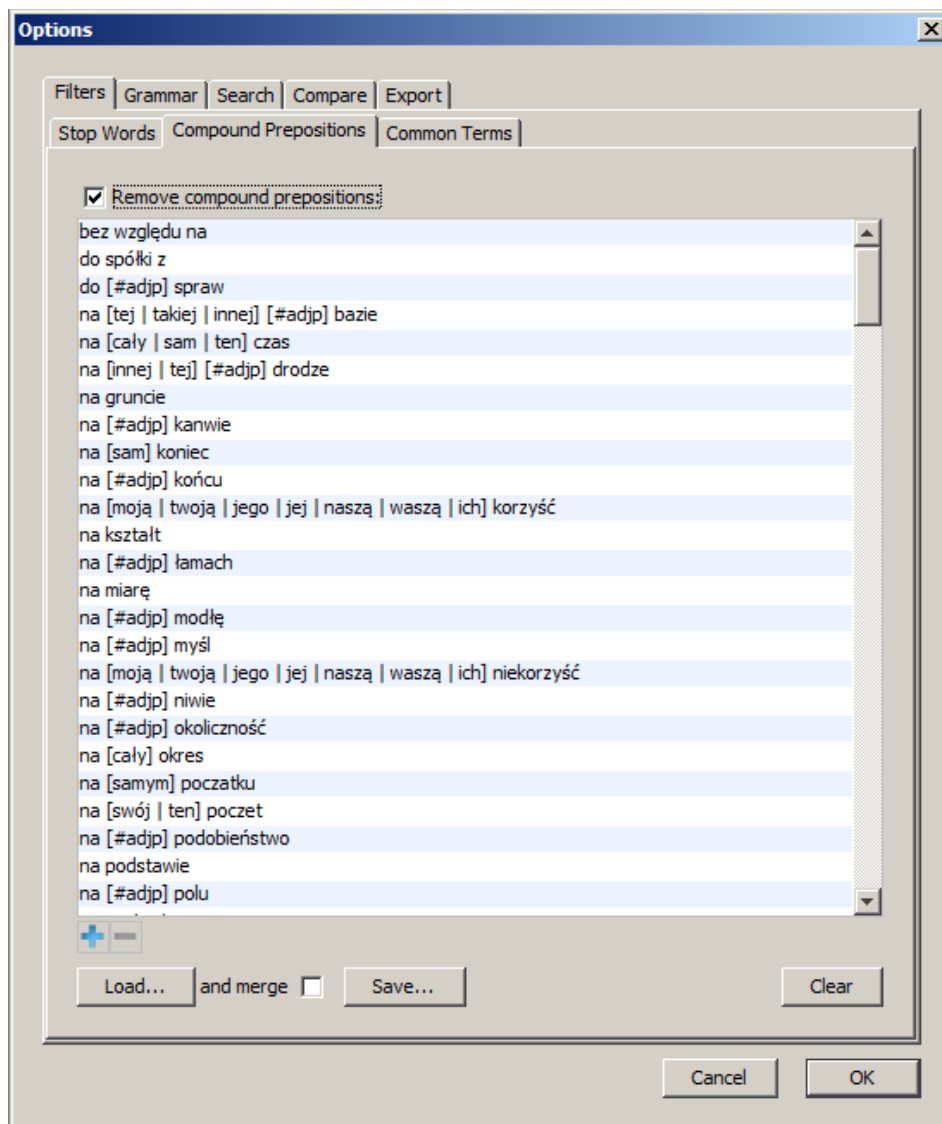


Figure 10: Options – Filters – Compound Prepositions.

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`.

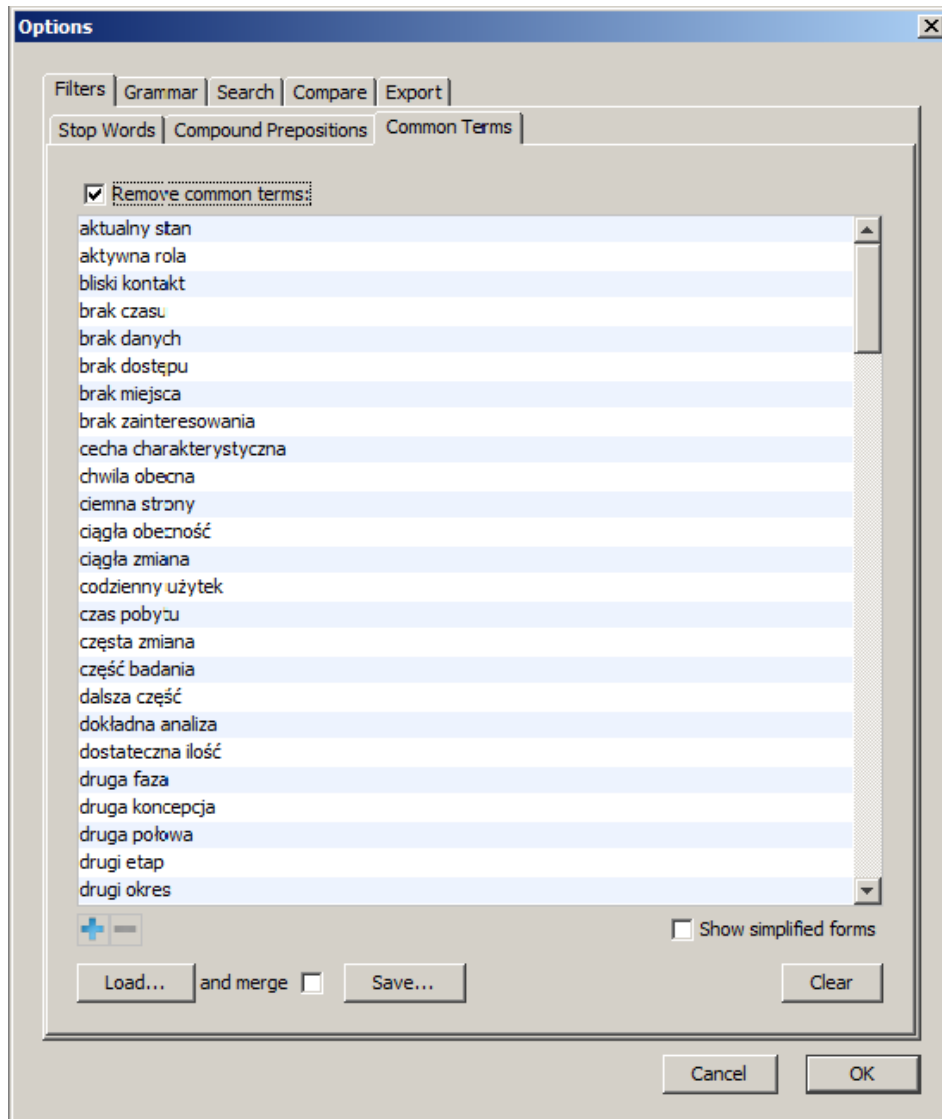


Figure 11: Options – Filters – Common Terms.

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.

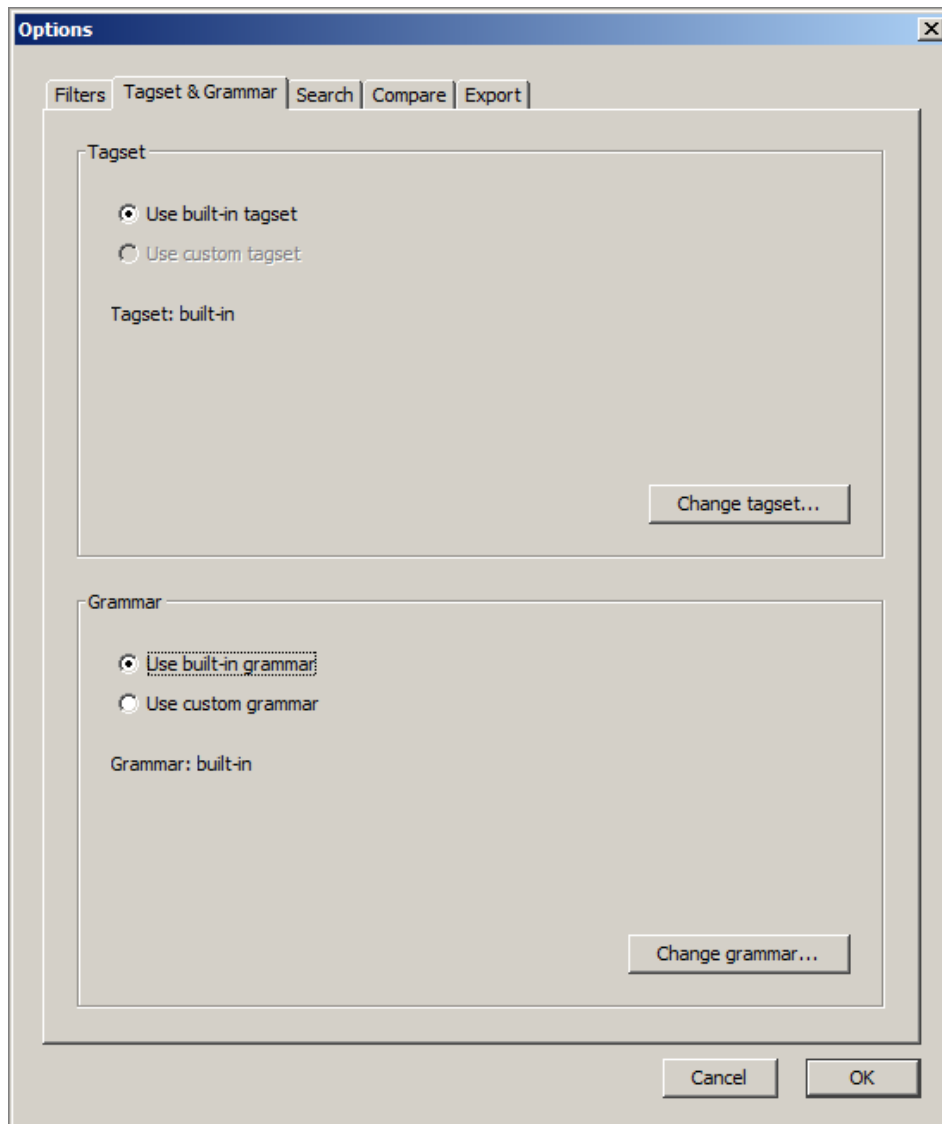


Figure 12: Options – Grammar.

Using this panel we can decide which grammar should be used by the program. 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.

Generating base forms of the extracted terms is possible only if the built-in tagset is selected. Grammar files as well as tagset files should be UTF-8 encoded text files.

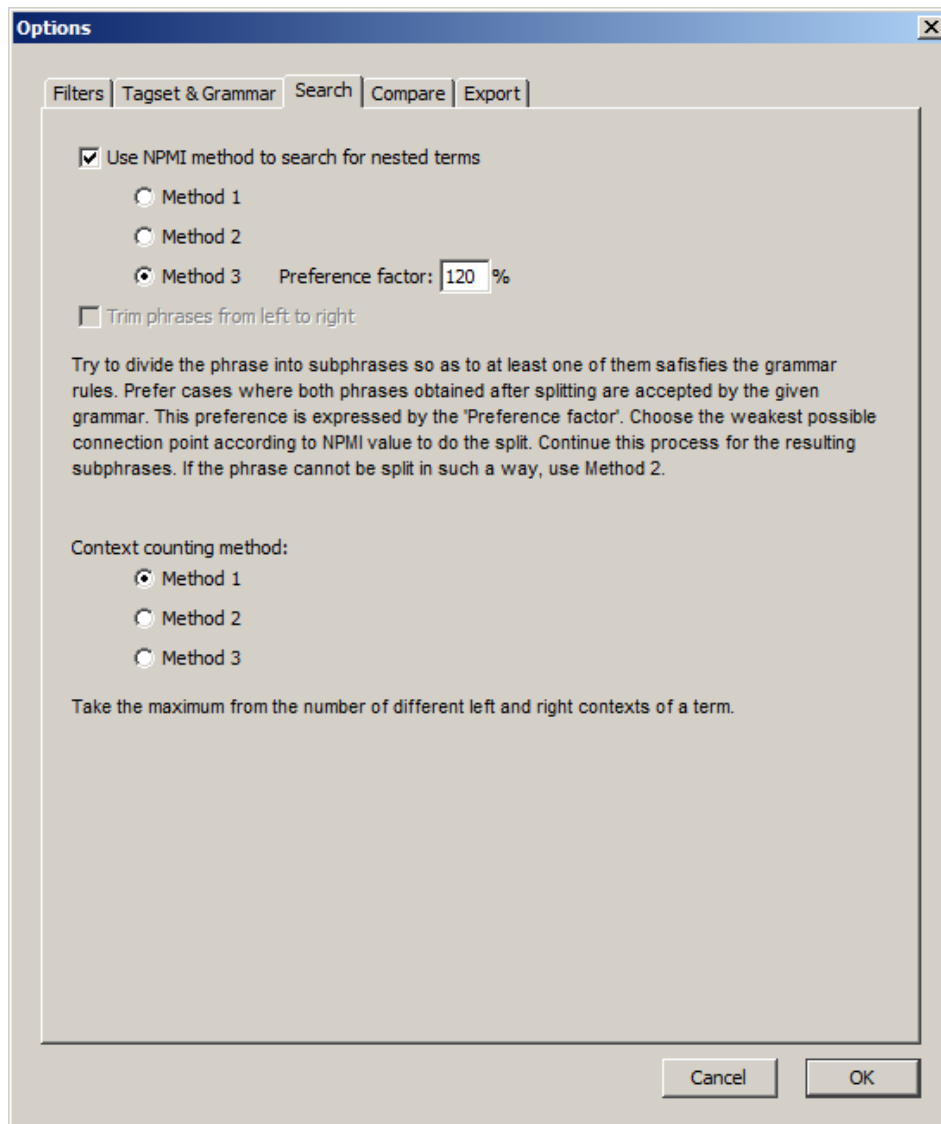


Figure 13: Options – Search.

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) .

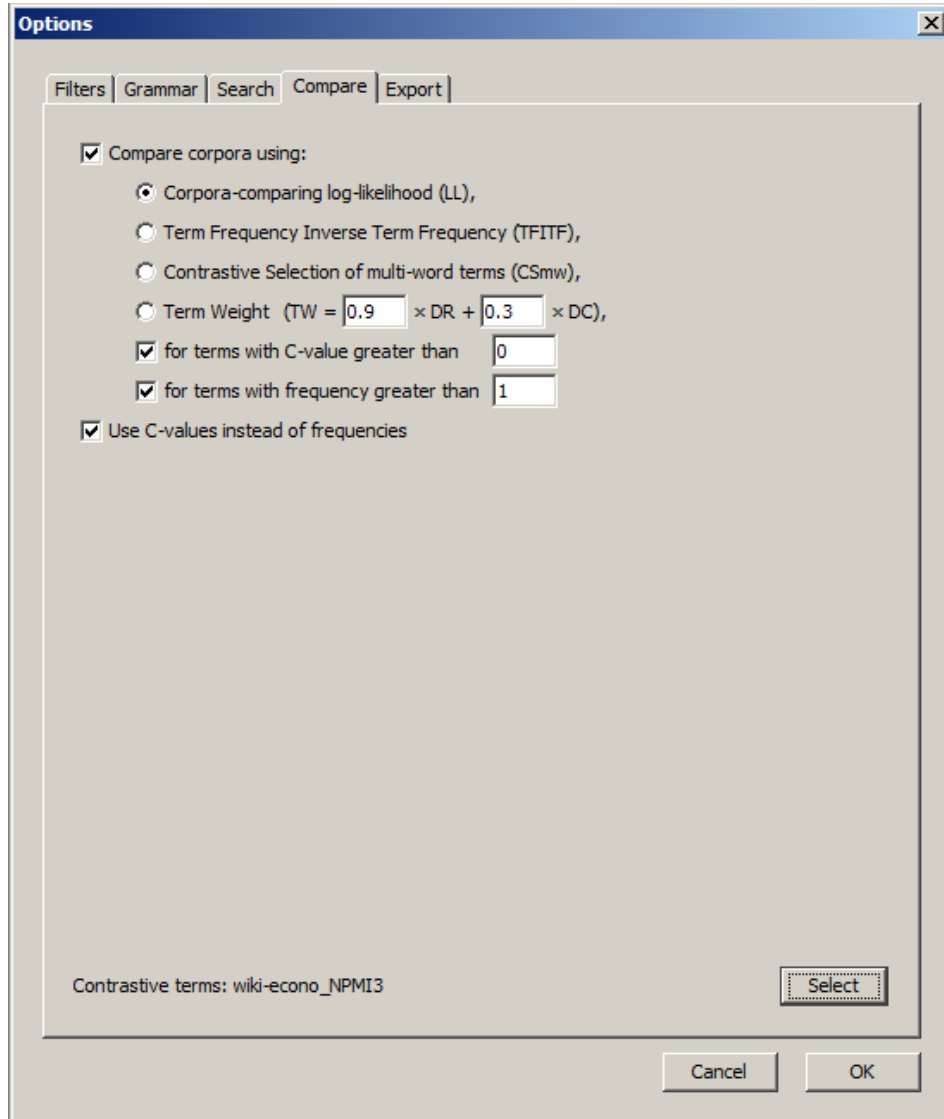


Figure 14: Options – Compare.

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

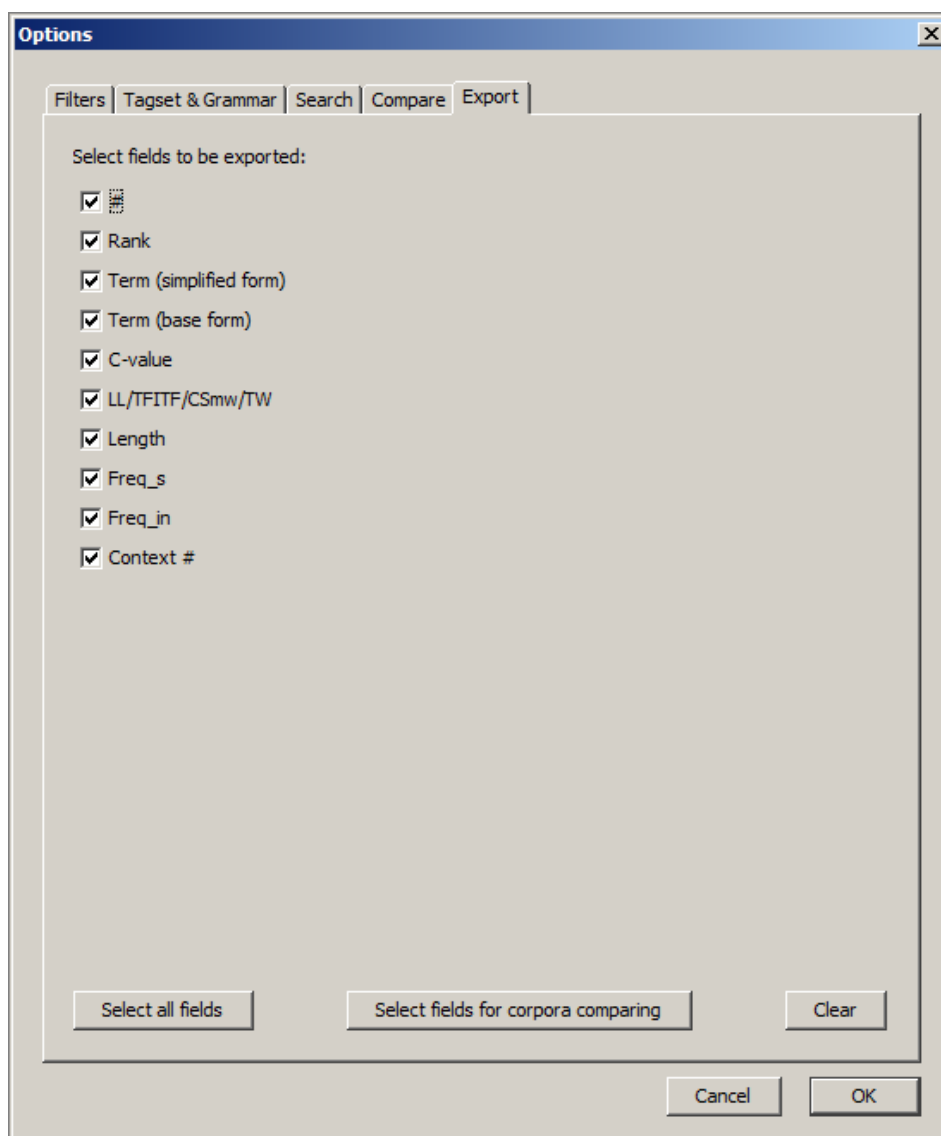


Figure 15: Options – Export.

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 15 shows default settings for the **Export** panel.

6.5 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.

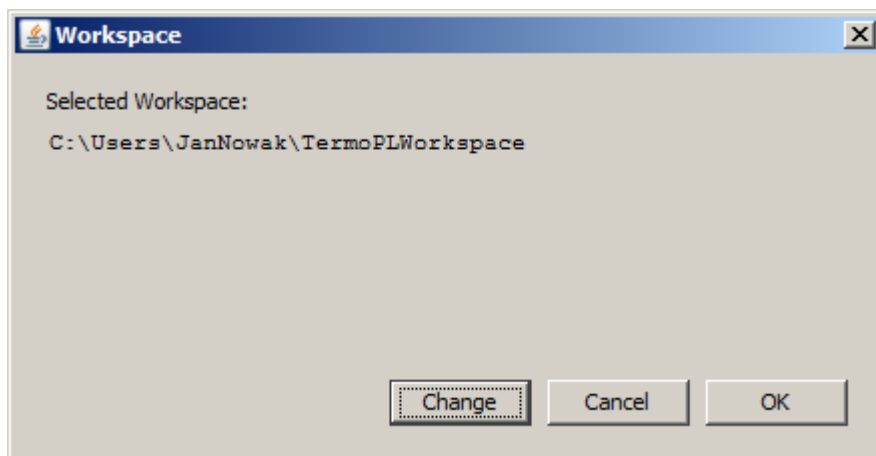


Figure 16: Workspace window

6.6 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.

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 20). 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 extracted terms (the **Export Forms...** command) and containing them sentences (the **Export Sentences...** command) to separate UTF-8 encoded text files. The user may choose to save all forms of extracted terms and containing

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

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.

6.7 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 19).

Choosing the **Select File(s)** command displays a dialog (Figure 17) 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 6.6). Removing any file from the old list causes that the extraction will be performed from scratch using all files from the list.

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 17).

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).

Once all necessary files are selected and all search options are set we can start the search. The process of extraction can be cancelled at any time.

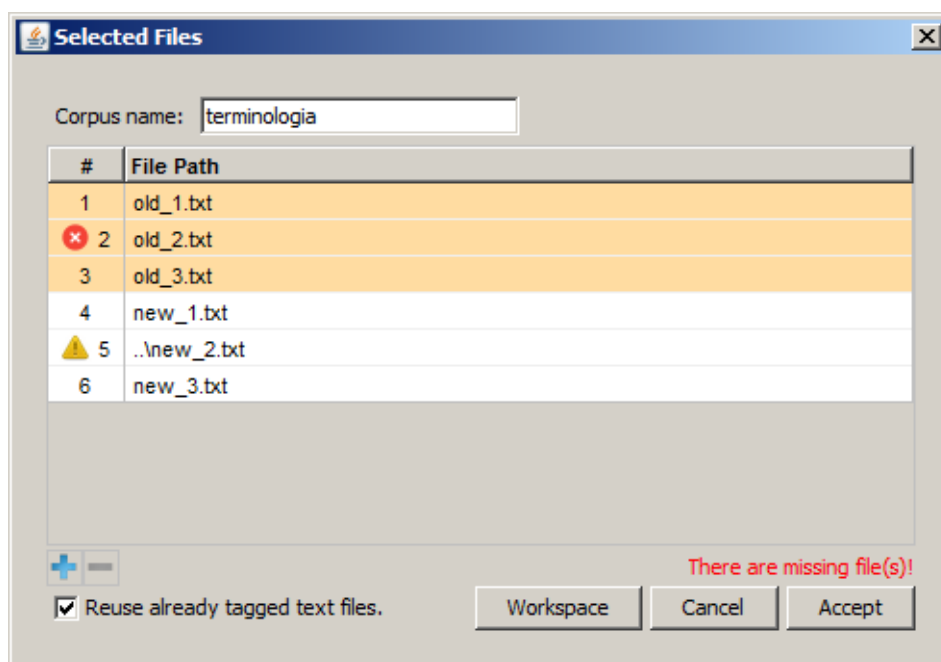


Figure 17: Selected Files

6.8 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), or view sentences from the corpus containing that term (the **Sentences with Selected Term** command). As it is explained in Section 6.7, 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).

6.9 The Window Menu

While working with TernoPL, 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 TernoPL windows one over another such that for all covered windows only the title bars are visible.

The **Tile** command organises all open TernoPL 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.

6.10 The Help Menu

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

7 Batch processing

TernoPL 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 TernoPL.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 TernoPL.jar [program options] file...
```

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

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

option	argumant(s) and meaning
-conf	configuration file (this option cannot be used in a configuration file)
-wrk	workspace
-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
-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
-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
-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). TernoPL 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 TernoPL is installed.

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

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

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

causes the program to run in the interactive mode.

8 Requirements

TernoPL is written in Java programming language and therefore requires Java Runtime Environment⁴(version 8 or later) to be installed on a target machine (Windows, Linux or Mac OS X). Since TernoPL uses Morfeusz 2.0⁵ 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.

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

⁴Java Runtime Environment can be downloaded from <http://www.java.com/pl/download/>

⁵Morfeusz 2.0 is available on <http://sgjp.pl/morfeusz/dopobrania.html>

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.

Using the Conraft tagger also requires more RAM.

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] 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.
- [4] 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.
- [5] 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).
- [6] Roberto Navigli and Paola Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics*, 30(2):151–179, 2004.
- [7] Adam Przepiórkowski, Mirosław Bańko, R. L. Górski, and Barbara Lewandowska-Tomaszczyk, editors. *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warszawa, 2012.
- [8] 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.
- [9] Jakub Waszczuk. Harnessing the CRF complexity with domain-specific constraints. The case of morphosyntactic tagging of a highly inflected language. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 2789–2804, Mumbai, India, 2012.

- [10] 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.