

TermoPL

user manual

Institute of Computer Science
Polish Academy of Sciences

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 recognize 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 assumes that the whole set of documents was first processed by a tagger. It accepts an UTF8 encoded input with morphosyntactic analysis in three different formats: NKJP [7], XCES, and the simple, flat format:

```
form #lemma #tag#,
```

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, all terminated by the # character. In this format, sentences are separated by a special line:

```
& #& #interp#.
```

If the input file contains more than one document from the analyzed corpus, they can be 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.

Termopl reads input sentence by sentence and identifies the maximal sequences of consecutive tokens that are recognized, 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 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

```

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.

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 recognized 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 recognize 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 romańska* 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.

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 [9, the morphosyntactic analyzer 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 colors according to their representativeness in analyzed 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.

3.2 Normalized pointwise mutual information

In order to determine the connection strength for a pair of words, TermoPL counts normalized 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:

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

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:

$$DR(t) = \frac{P_A(t)}{\max(P_A(t), P_B(t))},$$

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

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 Customizing 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> "]" ] : <regular expression over symbols> ";",
<symbol> "[" <test> "];".

```

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 a test or a series of tests performed during the matching process. 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 is a boolean function or 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 ('!=', '!'~') a test succeeds if it succeeds for all given strings.

Tests can be applied to single tokens or sequences of tokens. In the simple grammar given above, $N[pos = subst, ger]$ 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. Note however, 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.

There is only one boolean function *agreement* defined in ThermoPL and seven selectors whose names are self-explaining: *form*, *lemma*, *tag*, *pos*, *number*, *case* and *gender*. This set of methods can be fairly easy augmented and modified.

5 Graphical user interface

The graphical user interface layout is shown in Figure 2 where the candidate list is presented. Figure 3 shows the results of two corpora comparison.

#	▲ Rank	△ Term	▽ C-value	▽ Length	▽ Freq_s	▽ Freq_in	▽ Context #
1	1	papier wartościowy	281,21	2	284	187	67
2	2	działalność gospodarcza	217,45	2	220	135	53
3	3	cena	155,49	1	1558	1111	364
4	4	osoba fizyczna	154,52	2	156	34	23
5	5	podatek dochodowy	146,11	2	148	53	28
6	6	fundusz inwestycyjny	139,62	2	143	98	29
7	7	spółka	134,59	1	1350	966	233
8	8	rynek	131,22	1	1315	837	302
9	9	koszt	126,85	1	1271	1020	410
10	10	podatek	120,83	1	1212	756	206
11	11	wartość	114,56	1	1148	908	378
12	12	osoba prawna	111,17	2	113	33	18
13	13	kodeks spółek handlowych	110,95	3	71	5	5

Forms:
papierów wartościowych[26,137], papier wartościowy[6,5], papiery wartościowe[29,12], papierem wartościowym[4,0], papierami wartościowymi[23,2], papieru wartościowego[6,4], papierach wartościowych[3,0], Papierów Wartościowych[0,26], Papierów wartościowych[0,1]

Sentences: 17265; Tokens: 456195; Terms: 55033

Buttons: Open..., Save..., Options..., i, Cancel, Extract, Compare

Figure 2: The main window showing search results.

To perform a term extraction one has to load a corpus by selecting a file, a group of files, or a whole directory [*Open...*], set up the program options [*Options...*] and then press the button [*Extract*]. The process of extraction can be cancelled [*Cancel*] at any time. When the program finishes the extraction process it displays a table of term candidates. 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 first) column by clicking on its header. Columns may be sorted in ascending (▲) or descending (▼) order.

One may choose to collect all forms of extracted terms as they appear in an analyzed corpus (see page 11). In this case, selecting a row in the table will display all forms of the corresponding term on the bottom of the window. For each form, the number of its occurrences as an independent and nested phrase is given in square brackets.

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.

Once a set of terms is extracted from a corpus, it may be saved [Save...] to a file and subsequently used in corpora comparing.

To compare two corpora the user has to select one of the comparison methods (see page 12) and pick out one of the previously saved set of terms. Then he/she has to perform term extraction of domain corpus as it is described above. In case the domain corpora is already analyzed, it is sufficient to click on the [Compare] button.

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 3, log-likelihood values are shown.

#	Rank	Term	C-value	LL	Length	Freq_s	Freq_in	Context #
497	1222	góra	2,46	4,95	1	26	7	5
498	1398	pomysł	1,6	4,94	1	17	9	9
499	643	sektor publiczny	8,8	4,92	2	10	6	5
500	1447	żołnierz	1,39	4,89	1	15	9	8
501	647	proces produkcyjny	8,75	4,88	2	10	5	4
502	1576	krw	0,6	4,88	1	7	3	3
503	907	stanowisko	4,69	4,87	1	48	39	34
504	208	miejsce zamieszkania	26,8	4,86	2	28	12	10
505	560	wysoka jakość	10,6	4,85	2	12	7	5
506	508	wkład	12,2	4,85	1	124	85	42
507	436	szkoła	14,53	4,85	1	147	110	64
508	1567	wrażenie	0,68	4,84	1	8	5	4
509	1317	Marszałek Sejmu	2	4,83	2	3	1	1
510	440	kryterium	14,36	4,82	1	145	101	71
511	170	potrzeba	31,05	4,79	1	312	233	152
512	319	największy wpływ	20	4,79	2	20	0	0
513	565	obowiązujące przepisy	10,5	4,78	2	12	6	4

Figure 3: Results of two corpora comparison.

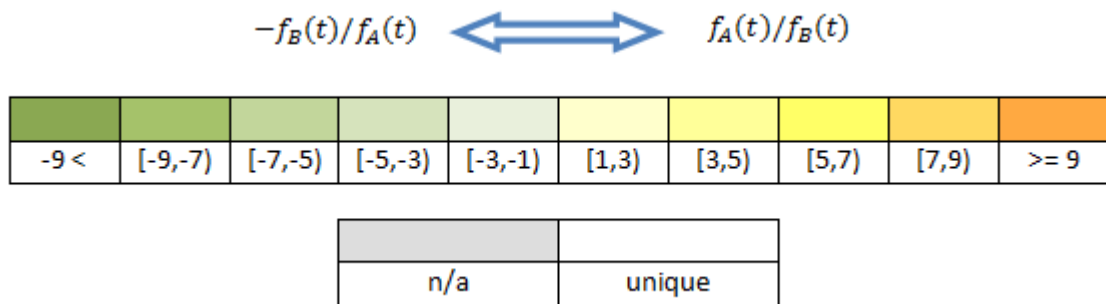


Figure 4: The meaning of table's colors.

The colors 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 colors show the opposite. The more saturated color, the more representative a given term is for one

of two corpora.

The user can change the behaviour of the program by setting different options [Options...]. In the interactive mode, the initial values for the options are loaded from the file `.TermoPL`, 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 Options dialog consists of six panels. In the first three panels we can define lists of filters: stop words (page 8), compound prepositions (page 9) and common terms (page 10). 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.

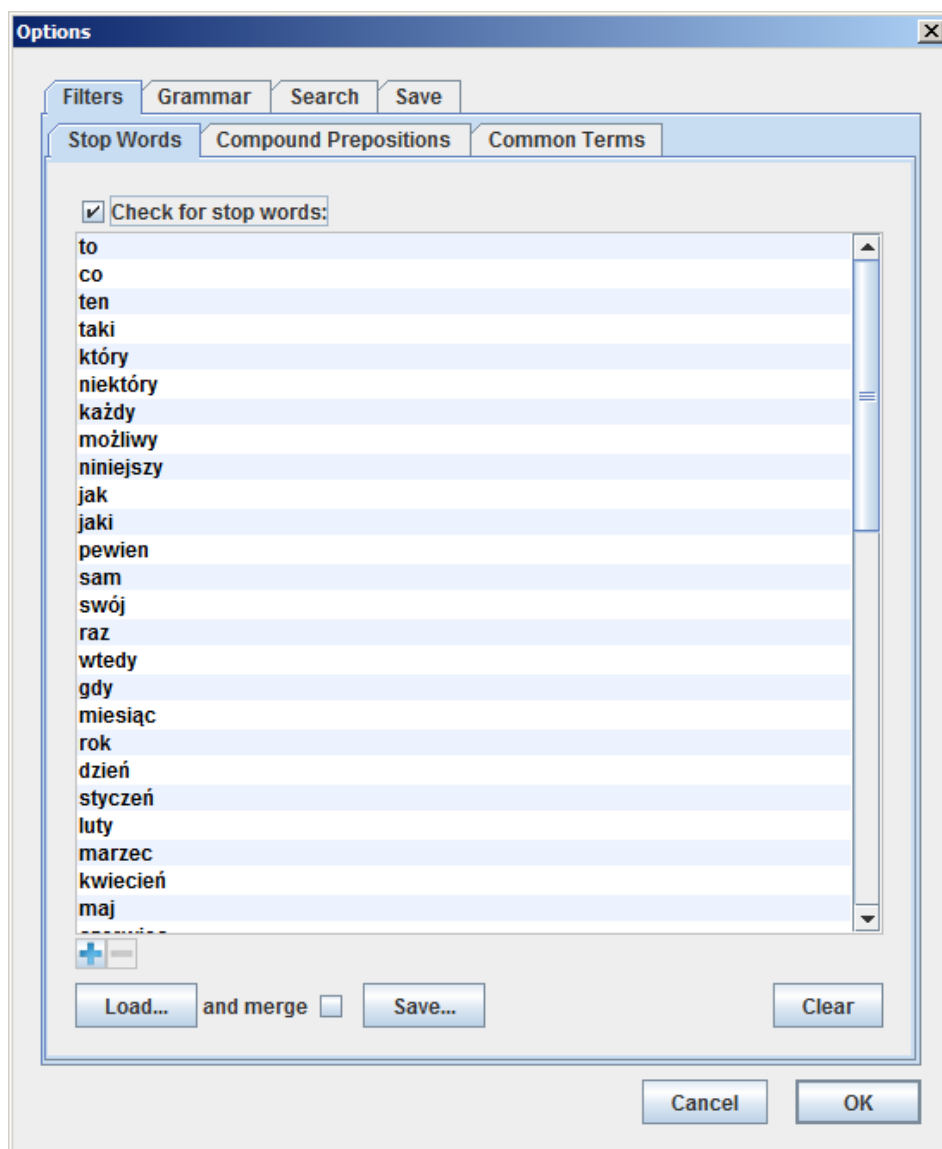


Figure 5: Optios – Filters – Stop Words.

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`.

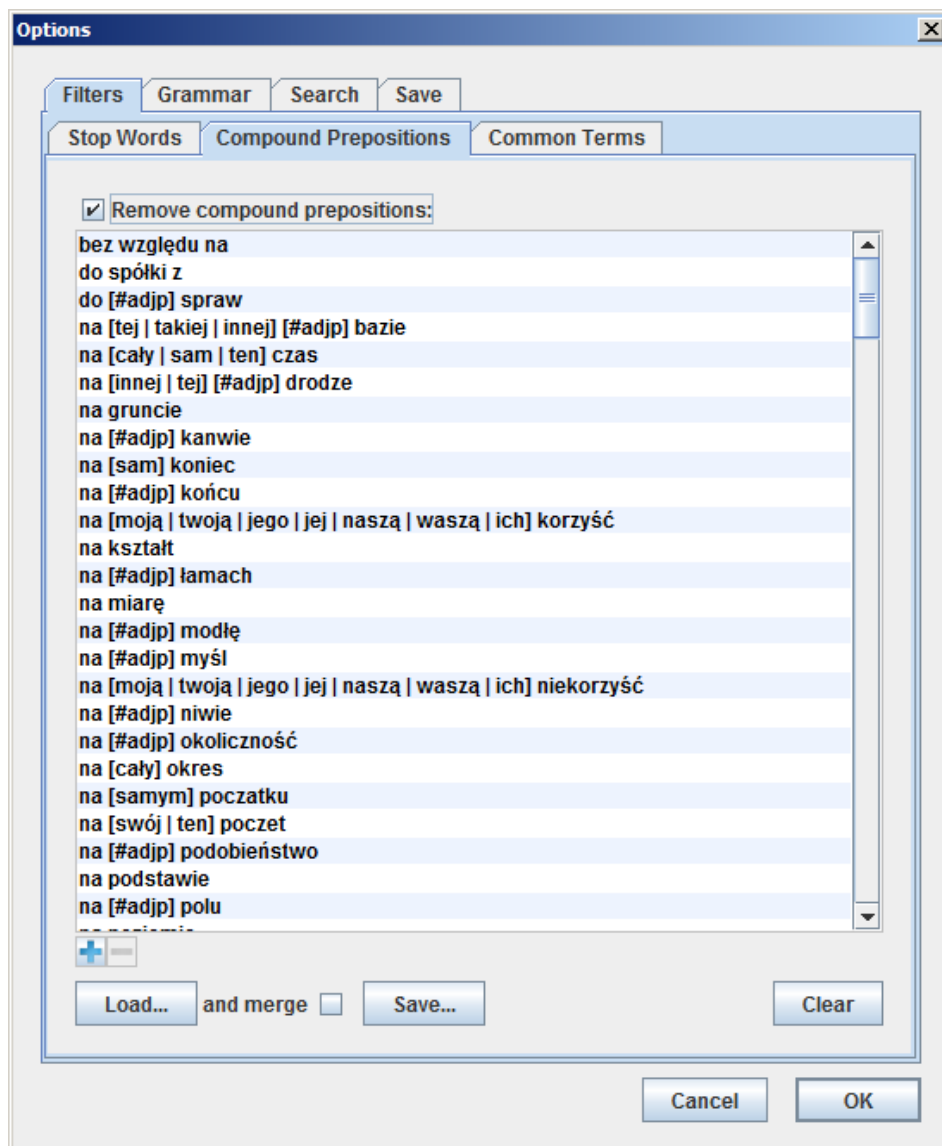


Figure 6: Options – Filters – Compound Prepositions.

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`.

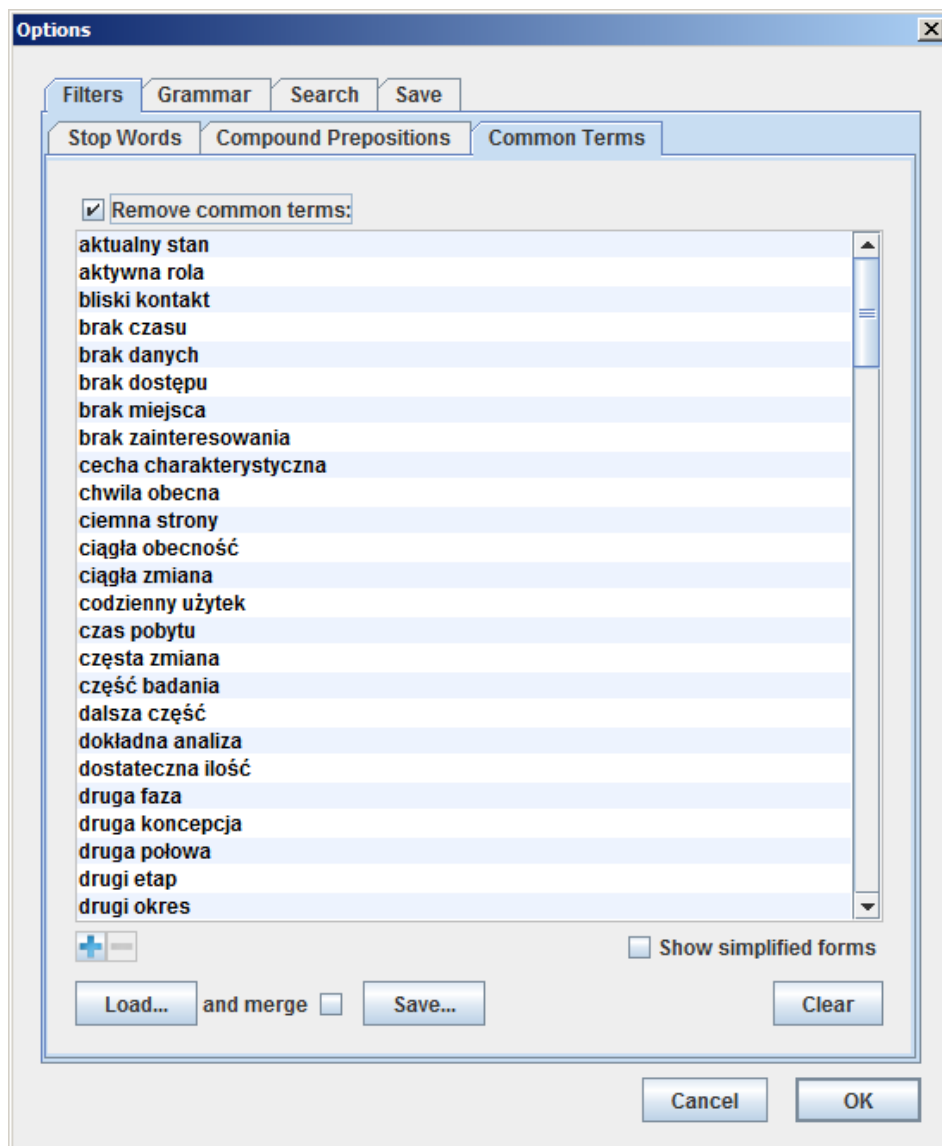


Figure 7: Optios – Filters – Common Terms.

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.

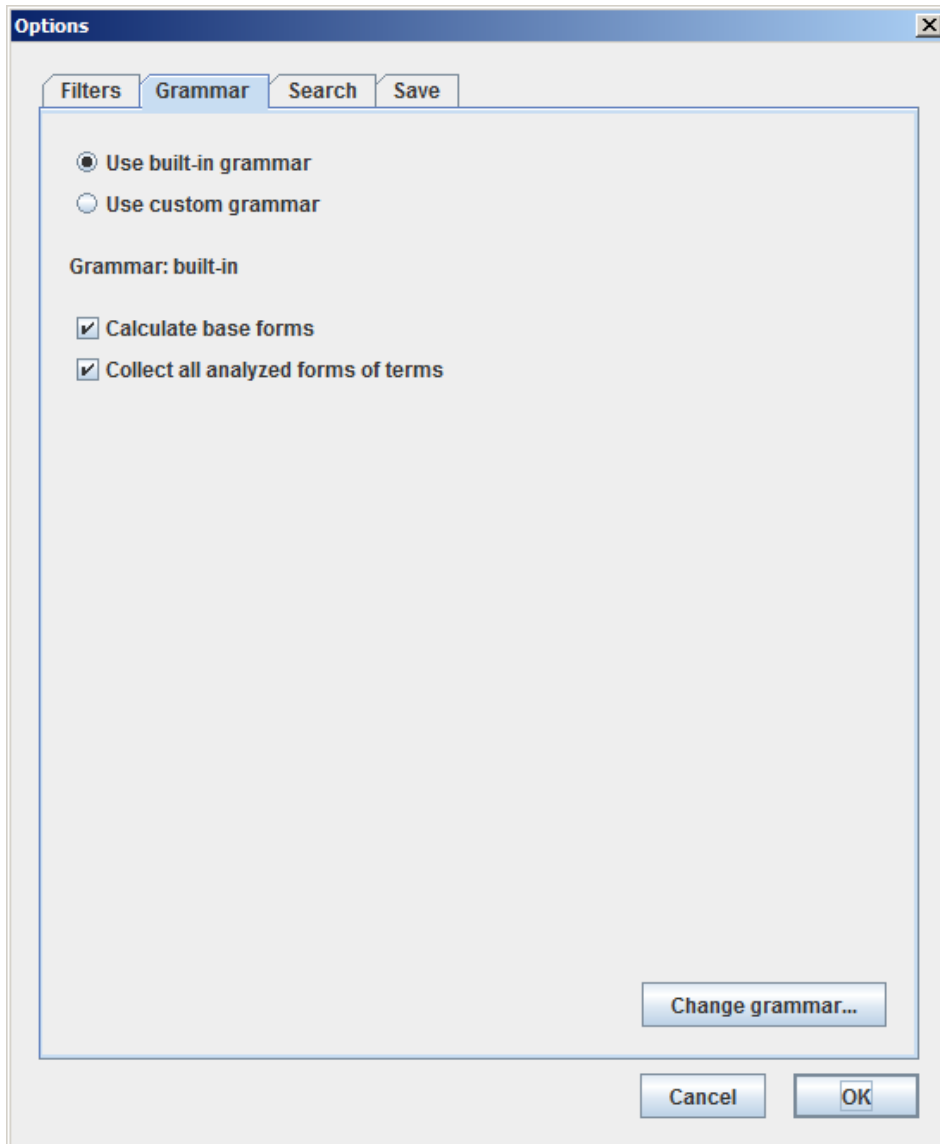


Figure 8: Options – Grammar.

Using this panel we can decide which grammar should be used by the program. Additionally, we can instruct the program to calculate base forms for extracted terms and collect all forms of terms in which they appear in an analyzed corpus. Calculating base forms requires the Morfeusz 2.0 libraries to be installed. By default the program uses the built-in grammar, calculates base forms and does not collect forms of terms.

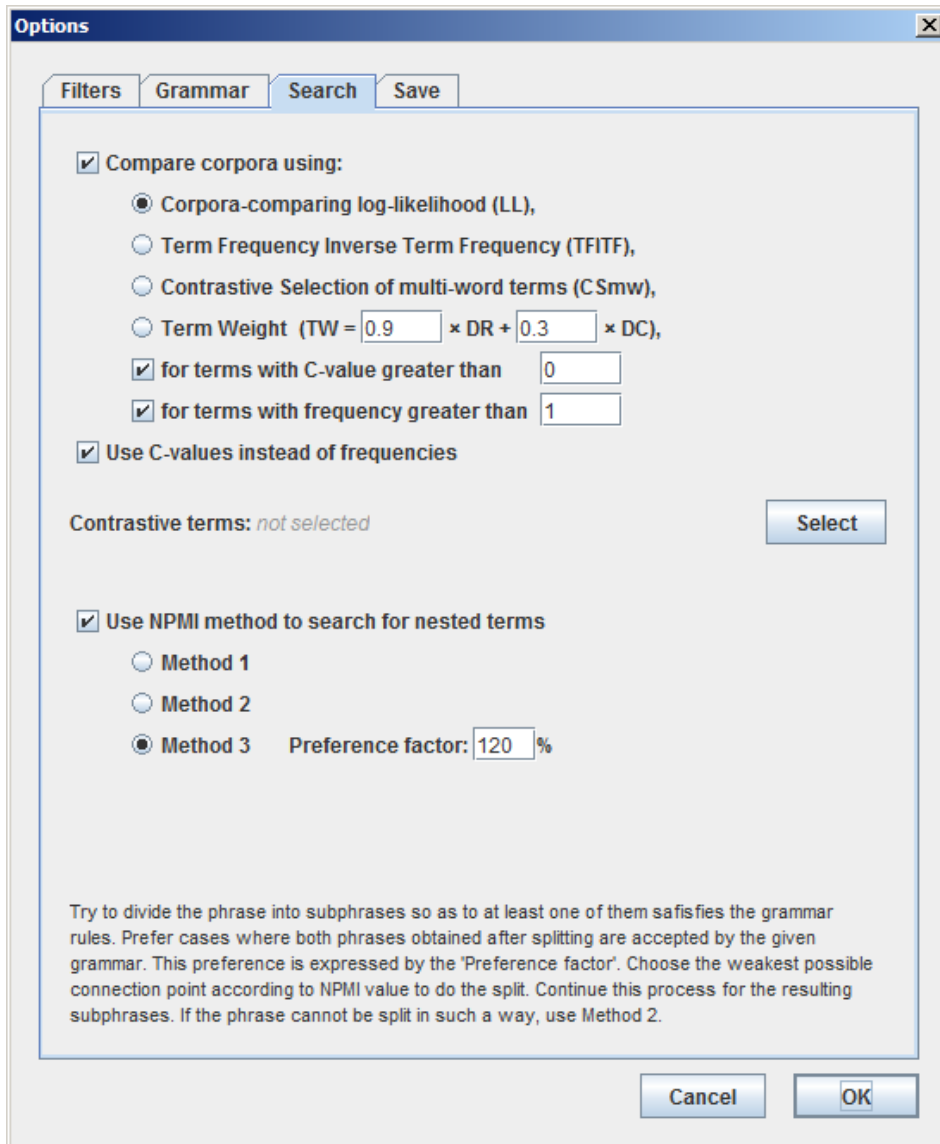


Figure 9: Options – Search.

The user should use this panel to set up the term extraction and corpora comparing methods, and their parameters. By default the program only performs the term extraction task using the third NPMI method. To run the corpora comparison task, the check box at the top of the panel must be selected and it is convenient then to choose [Select...] a contrastive set of terms. If a set of contrastive terms is not selected, the program will ask the user to do so.

In Figure 5 default settings for the Search panel are presented when we choose to compare corpora.



Figure 10: Options – Save.

The user can decide which columns from the results table will be saved to a 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 5 shows default settings for the **Save** panel.

6 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:

option	argumant(s) and meaning
-conf	configuration file (this option cannot be used in a configuration file)
-in	input file(s) (this option cannot be used in a command line)
-out	output file
-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
-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
-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
-nmpi	select search method; no – find all term candidates; 1 , 2 or 3 – use one of the NPMI 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: # , rank , sf (term's simplified form), bf (term's base form), cvalue , comp , length , freq_s , freq_in , context ; save_all_forms – save all forms of a term

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.

7 Requirements

`TermoPL` is written in Java programming language and therefore requires Java Runtime Environment¹ (version 7 or later) to be installed on a target machine (Windows, Linux or Mac OS X). Since `TermoPL` uses `Morfeusz 2.0`² 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.

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.

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,

¹Java Runtime Environment can be downloaded from <http://www.java.com/pl/download/>

²`Morfeusz 2.0` is available on <http://sgjp.pl/morfeusz/dopobrania.html>

- 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] 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.