Adam Przepiórkowski

The IPI PAN Corpus
preliminary version

INSTITUTE OF COMPUTER SCIENCE PAS
WARSAW 2004

Institute of Computer Science
Polish Academy of Sciences
ul. Ordona 21
01-237 Warszawa
Poland

ISBN 83-910948-8-X

Contents

1

Introduction

1.1. The IPI PAN Corpus

This publication is an outcome of a project financed chiefly by the State Committee for Scientific Research (Komitet Badań Naukowych; KBN; project number 7 T11C 043 20) carried out at the Institute of Computer Science, Polish Academy of Sciences (ICS PAS) between April 2001 and March 2004, as well as the result of statutory research carried out at ICS PAS. Its aim is to present some of the results of the project, namely, the IPI PAN[1] Corpus of written Polish, as well as tools for searching the corpus. This presentation should make it possible to effectively use both the corpus and the tools.

To the best of our knowledge, the IPI PAN Corpus is the first publicly accessible corpus of Polish, where the term *corpus* is understood here as a large (over 100 mln. running words) linguistically (morphosyntactically) annotated collection of texts, containing a balanced subcorpus, developed in accordance with current standards and best practices in so-called corpus linguistics.

Such corpora exist for numerous languages, not only European, and they are widely used in Natural Language Processing (NLP), in lexicography, and in other subfields of linguistics. Many countries consider the creation of such a corpus a duty towards the languages spoken there,

[1] The name of the corpus should be pronounced as in "E.P. Pahn Corpus", IPI PAN being the abbreviation of *Instytut Podstaw Informatyki, Polska Akademia Nauk*, the Polish name of ICS PAS.

hence the development of 'national' corpora such the British National Corpus (http://www.hcu.ox.ac.uk/BNC/), or the Czech National Corpus (http://ucnk.ff.cuni.cz/). So far, the only publicly available linguistically annotated corpus of Polish has been the corpus of the "Frequency dictionary of contemporary Polish" (Kurcz *et al.*, 1974, 1990), containing around half a million running words.

Previous corpus research in Poland has been scattered over a number of sites, and only very limited results of such work are publicly available at the time of writing this publication. Corpus research has been carried out, *inter alia*, in Warsaw (the PWN publishing house), Cracow (Institute of the Polish Language, Polish Academy of Sciences; IPL PAS), Łódź (University of Łódź) and in Wrocław (University of Wrocław). The currently available results of that research are raw (not linguistically annotated) samples of the PWN corpus, i.e., a 2 million sample available for searching at http://korpus.pwn.pl/, another sample, four times as large, sold with the luxury edition of PWN's "Universal dictionary of Polish", as well as a sample sold by the University of Łódź, containing 10 million running words. The aim of the current project has been to address the problem of public unavailability of extensive corpus data from Polish and thus to provide the basis for wider development of statistical NLP methods in Poland.

The binary version of the IPI PAN Corpus, distributed on the CD-ROM enclosed with this publication, is aimed, first of all, at linguists and other linguistically conscious speakers and learners of Polish. This binary corpus should be accessed via the search tool Poliqarp, also enclosed on the CD-ROM. The sources of the corpus, probably more useful for various NLP applications, are available directly from ICS PAS (inquiries should be directed to info@korpus.pl or adamp@ipipan.waw.pl).

The current version of both the corpus and the tools is called here a *preliminary version*: we are painfully aware of various inadequacies of the corpus and the tools available on the enclosed CD-ROM. Taking into consideration the sheer size of the corpus, and the limited resources at the disposal of the project, it was impossible to verify the results of the automatic conversion of the incoming texts into the XML format, or the results of morphosyntactic, structural or metadata annotation. Especially the last kind of information, the general information about texts (their origin, title, author, etc.), should be considered as incomplete and extremely

preliminary. The IPI PAN Corpus in its current form is a typical *opportunistic* corpus, containing various genres in unbalanced proportions. A more careful selection of a fully balanced subcorpus is a task which should be addressed at the next stage of corpus development.

Also the corpus search tool enclosed on the CD-ROM, Poliqarp, even though it can in many respects be favourably compared to other available corpus search tools (e.g., CQP, GCQP, Bonito), is far from constituting a finished product: the efficiency of searching of corpora over 50 million words is still too low, virtually no statistical functionality is currently available, there is no possibility of building queries the graphical way, and the user's influence on the output format is limited. Those and other inadequacies will be addressed in future releases of the tool.

1.2. Acknowledgements

The IPI PAN Corpus and various tools for corpus creation and access have been developed chiefly within the State Committee for Scientific Research project, as well as within statutory research carried out at the Institute of Computer Science, Polish Academy of Sciences, so it would be a truism to say that without the support of these two institutions this project would have never taken off the ground.

The success of this project owes much to the support of various people and institutions. Prof. Zygmunt Saloni (University of Warmia and Mazury in Olsztyn) and Marcin Woliński (ICS PAS) provided the project with a morphological analyser, *Morfeusz*. Prof. Janusz S. Bień gave us access to a cleaned-up version of the corpus of the "Frequency dictionary of contemporary Polish", mentioned above. Dr. Jan Hajič (Charles University, Prague) put at our disposal a tool for the manual disambiguation of morphosyntactic forms, DAUJC, developed by Jiří Hana. Prof. Włodzimierz Gruszczyński (University of Warsaw; UW) made available to the project the numerous nominal paradigms, which helped to improve the results of the morphosyntactic annotation of the corpus. Prof. František Čermak (Institute of the Czech National Corpus, Charles University, Prague) invited the staff of the IPI PAN Corpus project to pay a visit at the Institute of the Czech National Corpus and learn from the experience of our Czech colleagues.

One of the most time consuming and tedious tasks of the project was the acquisition of texts and appropriate copyrights for publishing the texts in a publicly available corpus. The process of acquisition was run chiefly by Dr. Rafał L. Górski (IPL PAS). What helped us convince authors and publishers to the idea of a large publicly available corpus of Polish were the recommendations written for us by Prof. Jerzy Bralczyk (UW), Prof. Stanisław Gajda (the Committee of Linguistics, Polish Academy of Sciences), and Prof. Ireneusz Bobrowski (IPL PAS). The list of people who helped us reach authors and publishers is too long to be included here (but see the WWW pages of the project).

As the leader and principal investigator of the project I would like to cordially thank all investigators of the project for their hard work and commitment. Łukasz Dębowski (ICS PAS), apart from being responsible for the statistical tagger used in the project, also provided help in managing the project. The detailed morphosyntactic tagset developed within the project is the result of numerous discussions between the author, Łukasz Dębowski and Marcin Woliński, as well as Elżbieta Hajnicz (ICS PAS) and Zygmunt Saloni. Marcin Woliński also created a tool for the effective extension of the empirical scope of the morphological analyser used in the project. The manual disambiguation of the training corpus was performed by Monika Czerepowicka (University of Warmia and Mazury in Olsztyn), Dorota Lewandowska (UW), Hanna Maliszewska (UW), Marta Nazarczuk-Błońska, Marta Piasecka (UW), Beata Wójtowicz (UW) and Ewa Wolska; they also provided valuable feedback on the tagset. The quality of that manual disambiguation was controlled by Elżbieta Hajnicz.

The most difficult and time-consuming programming task within the project was the development of Poliqarp, the tool for indexing, searching and concordancing. The main authors of that tool, whose functionality in some respects exceeds the state of the art, are Zygmunt Krynicki (Polish-Japanese Institute of Information Technology) and Daniel Janus (UW).

Another difficult and tedious task was the conversion of texts received from authors and publishers into a uniform XML format, whose initial versions were designed by Dr. Piotr Bański (UW). Investigators involved in the creation of programs supporting the conversion, as well as in the actual conversion, include: Piotr Bański, Artur Gniadzik (UW), Paweł Savov

(UW), Katarzyna Sokołowska (UW), Radosław Moszczyński (UW), Jakub Sikora (UW) and Jakub Jurkiewicz (UW).

It is my hope that the cause for a large linguistically annotated and publicly available corpus of Polish will continue to attract the support of many people and institutions.

2

Preliminary text processing

Texts included in the IPI PAN Corpus undergo a long process of transformation from the form in which they are acquired to the form accessed via the search and concordance tool described in ch. 4. The present chapter briefly discusses the stages of this conversion process.

2.1. From the original format to XML

All texts acquired within the project are converted from their original format, e.g., HTML, Word, RTF, PDF, WordPerfect, LaTeX, etc., into a uniform open textual format. For the purposes of the current project, that common format is the slightly modified *XML Corpus Encoding Standard* (XCES; Ide *et al.* 2000). XCES itself is an XML version of an earlier SGML standard, the *Corpus Encoding Standard* (CES; Ide *et al.* 1996), based on the *Text Encoding Initiative* (TEI) recommendations. Also character encoding is converted to the common universal character set, i.e., to UTF-8.

Each text included in the IPI PAN Corpus is converted to three XML files, located in a separate directory (folder):[1]

- `header.xml`: contains the metadata, i.e., data concerning people or institutions responsible for the creation of the text, normally its author(s), the title, the publisher, dates of creation and publication, as

[1] The initial stages of the adaptation of the XCES standard to the needs of the current project are presented in Bański 2001, 2003.

well as information about the process of converting the texts into XML and about further modifications in the results of this conversion;

- `text.xml`: contains information about structural divisions within the text (into chapters, paragraphs, etc.), as well as some typesetting information (especially, about italics, bold font, etc.);
- `morph.xml`: contains morphosyntactically annotated text, divided into sentences, paragraphs and some larger chunks of text.

Among the tools created within the project are tools supporting the process of conversion of Word, HTML and PDF documents into preliminary versions of `header.xml` and `text.xml` — texts in other formats are initially converted to one of these three formats with the aid of publicly available tools, or they are processed individually. Obviously, fully automatic extraction of correct information about the origin of texts and about their logical structure is impossible in case of documents containing only or mostly typesetting information, such as Word documents or PDF files, so post-conversion manual verification of these files is necessary. Such verification includes the deletion of foreign passages and other fragments of the original which do not represent continuous Polish texts, e.g., mathematical and chemical formulae. Apart from such deletions, texts are not normalised: numbers written in digits, including dates, are not translated into word forms, abbreviations are not expanded, errors are not corrected.

Because of the large amount of texts in the corpus, there were more than five people involved in the conversion process, with differing levels of computer and XML expertise. For this reason, despite the guidelines (Przepiórkowski, 2004) describing the mark-up in `header.xml` i `text.xml`, with special emphasis on the differences between the standard XCES and the modified version of XCES assumed in the project, certain cross-convertor differences were inevitable.

This first stage of text processing is the most time-consuming and labour-intensive stage in the conversion process. The results of this stage are the final version of `header.xml` and a preliminary version of `text.xml`. These files are valid XML files, validated with the version of XCES (`xcesDoc.dtd` and `xheader.elt`) assumed in this project.

2.2. Further XML processing

Further stages of text processing do not normally require human intervention. The preliminary version of `text.xml` created in the previous stage is converted into `morph.xml`, which does not contain detailed structural information, but contains the text divided into sentences and morphosyntactically annotated (cf. ch. 3).

Splitting the text into sentences is performed according to a simple algorithm which, for each potential sentence-final punctuation mark, investigates the context of that mark, in particular, whether that punctuation mark is a part of an abbreviation, and if it is, whether that abbreviation is a potentially sentence-final abbreviation, whether the next non-space character is a small letter or a capital letter, etc. Obviously, it is not always possible to decide whether a given potential sentence-final punctuation mark is an actual sign of the end of a sentence. The following Polish sentences illustrate that point.

(2.1) Kiedy to się działo? W latach 40. Stany Zjednoczone włączyły
when this Refl happened in years 40 States United entered
się do wojny.
Refl to war

When was that happening? In the forties, the United States entered the war.

When was that happening? In the forties. The United States entered the war (then).

(2.2) Skorzystać z Yahoo! Marek i jego koledzy nie chcieli.
use from Yahoo! Marek and his colleagues not wanted

As for using Yahoo!, Marek and his colleagues didn't want to do that.

To use Yahoo! (But) Marek and his colleagues didn't want to do that.

The text in `morph.xml` is not only divided into sentences, but also, further, into smaller segments (approximately, words) which are the units of morphosyntactic annotation, i.e., which are assigned to grammatical classes (so-called parts of speech) and which are assigned the values of appropriate grammatical categories such as case or person.

Morphosyntactic annotation is itself performed in two steps. First of all, the morphological analyser splits the text into tokens, or segments, and assigns to each segment (roughly, to each word, but see ch. 3 on why segments are not always words) all possible morphosyntactic interpretations, without any attempt at determining which of these interpretations are correct in the given context. The morphological analyser used in the current project is *Morfeusz*, developed by Marcin Woliński on the basis of linguistic data provided by Zygmunt Saloni, especially his database of Polish verbs (Saloni, 2001) and the stemming rules published as Tokarski 1993. The analyser is still under development, and many of the annotation errors in the current version of the IPI PAN Corpus stem from the inadequacies of the current version of the analyser.

The second step in the morphosyntactic annotation is the disambiguation of morphosyntactic interpretations provided by the morphological analyser, i.e., the selection of those interpretations which are appropriate in a given context. This disambiguation of morphosyntactic interpretations is performed by a program developed by Łukasz Dębowski, based on statistical disambiguation methods (cf. Dębowski 2001, 2003, 2004). The example below presents a small fragment of `morph.xml`, corresponding to the sequence of forms *Porządek dzienny*, 'daily order', lit. 'order daily'.[2]

```
<tok>
 <orth>Porządek</orth>
 <lex><base>porządek</base><ctag>subst:sg:acc:m3</ctag></lex>
 <lex disamb="1">
  <base>porządek</base><ctag>subst:sg:nom:m3</ctag>
 </lex>
</tok>
<tok>
 <orth>dzienny</orth>
 <lex><base>dzienny</base><ctag>adj:sg:acc:m3:pos</ctag></lex>
 <lex><base>dzienny</base><ctag>adj:sg:nom:m1:pos</ctag></lex>
 <lex><base>dzienny</base><ctag>adj:sg:nom:m2:pos</ctag></lex>
 <lex disamb="1">
  <base>dzienny</base><ctag>adj:sg:nom:m3:pos</ctag>
 </lex>
</tok>
```

[2] The meaning of tags such as `subst:sg:acc:m3` is explained in detail in ch. 3.

As this example shows, `morph.xml` contains not only the interpretations selected by the disambiguator (cf. '`disamb="1"`' above), but also all other interpretations originally assigned by the morphological analyser.

Interpretations of both forms occurring in the example above, i.e., *Porządek* and *dzienny*, are fully disambiguated (to singular, nominative, masculine inanimate). However, there are situations where the full disambiguation, to just one interpretation, would have to be utterly arbitrary, as in (2.3), involving a syncretic accusative / genitive pronoun and a verb taking an accusative or a genitive complement, where it is not possible to determine whether the form *go* 'him' occurs in the accusative (as in (2.4a)), or in the genitive (cf. (2.4b)).

(2.3) Pożądała go.
desired.FEM him.ACC/GEN
'She desired him.'

(2.4) a. Pożądał ją.
desired.MASC her.ACC
'He desired her.'

b. Pożądał jej.
desired.MASC her.GEN

Another example illustrating the same point is (2.5), where the form *pijaną* 'drunk' may be interpreted as accusative (as in (2.6a)) or as instrumental (cf. (2.6b)).

(2.5) Pamiętam ją pijaną.
remember.1ST her.ACC drunk.ACC/INS
'I remember her drunk.'

(2.6) a. Pamiętam go pijanego.
remember.1ST him.ACC drunk.ACC
'I remember him drunk.'

b. Pamiętam go pijanym.
remember.1ST him.ACC drunk.INS

In such cases, both interpretations (`<lex>` elements) should be marked as 'disambiguated' (`disamb="1"`).

The result of this stage of processing is the creation of the final version of `text.xml`, containing identifiers of all XML elements, as well as the final version of `morph.xml`, containing the morphosyntactic annotation and aligned with `text.xml` via references to those identifiers. Both files are valid XML files satisfying the slightly modified version of the XCES standard used in the project (`xheader.elt` and, respectively, `xcesDoc.dtd` and `xcesAna.dtd`).

2.3. From XML to binaries

Given the size of the corpus, searching directly in the XML files created within the previous stages of text processing would be extremely inefficient. For this reason, all `header.xml` and `morph.xml` files which constitute the corpus are compiled to a binary form, consisting of various indices which allow Poliqarp, the program described in ch. 4, to access any part of the corpus in an efficient way. During this compilation process, some of the information contained in headers (i.e., in the `header.xml` files) is ignored, but the most important information about the author, the title and the publication date, as well as complete morphosyntactic information, is retained and can be accessed via Poliqarp.

Only this binary representation of the corpus is available on the CD-ROM enclosed with this publication.

3

Tagset

The IPI PAN Corpus is annotated with morphosyntactic information. What that means is that sequences of characters (roughly, words) are assigned so-called morphosyntactic tags which interpret those sequences as certain word forms. We will call such interpretable sequences of characters *segments*. Segmentation rules applied in this corpus are described in §3.1.

One (or more, in some cases) of the interpretations assigned to a given segment is selected by the automatic disambiguator or by a human annotator as correct in the given context. For example, in case of the segment *nie*, the morphological analyser will assign to this segment, regardless of the context in which it occurs, one tag interpreting this segment as the negative particle NIE, and a series of tags corresponding to various pronominal interpretations of this segment. In case of *nie* occurring in the sequence *Janek nie przyszedł* 'John didn't come', lit. 'John not came', the first tag, corresponding to the interpretation of *nie* as the negative particle,

will be selected as correct in this context. On the other hand, in case of the sequence *Twoje koleżanki przyjdą, poczekaj na nie* 'Your friends will come, just wait for them', lit. 'Your friends.FEM come.FUT, wait for them', the accusative feminine plural post-prepositional pronominal interpretation will be selected as correct.

The IPI PAN Corpus contains both kinds of interpretations: all possible interpretations assigned to a given segment by the morphological analyser, as well as appropriately marked interpretations selected as correct in the given context. The internal structure of morphosyntactic tags is discussed in §3.2, while the complete repertoire of grammatical categories and classes adopted here is presented in §3.3 and §3.4.

The tagset discussed below is based on a rich body of work on Polish morphosyntax by Zygmunt Saloni and his colleagues (Saloni, 1976, 1977, 1981, 1988; Gruszczyński and Saloni, 1978; Bień and Saloni, 1982; Bień, 1991). The specification of the complete tagset and the segmentation rules was developed by Marcin Woliński and the author, and has been greatly influenced by many discussions with Łukasz Dębowski, Elżbieta Hajnicz and — in the final stages of research — Zygmunt Saloni. Previous versions of the tagset were presented and justified in Woliński and Przepiórkowski 2001, Przepiórkowski and Woliński 2003a,b, Woliński 2003 and Przepiórkowski 2003b, and in the guidelines for annotators (Przepiórkowski *et al.*, 2004a).

3.1. Text segmentation

Text segmentation consists in partitioning the text into sequences of characters which are subject to morphosyntactic interpretation, i.e., into segments. Segmentation rules are, even if often implicitly, part and parcel of the design of any tagset. One tagset is needed when forms of inherently reflexive verbs are split into two segments, e.g., in case *bał się* 'feared', lit. 'feared Refl' is split into *bał* and *się*, and another tagset is needed when such sequences are assigned only a single tag. Similarly, different tagsets are needed in case the word *przyszlibyśmy* 'we would come' is split into the segments *przyszli* 'come', *by* (subjunctive particle) and *śmy* 'Aux-1.PL', and in case that word is considered to be a single segment.

The fundamental segmentation principle adopted in the IPI PAN Corpus is as follows:

- segments are contiguous, i.e., they consist of a continuous sequence of characters, without gaps or other intervening segments, and
- they are disjoint, i.e., there are no characters which simultaneously belong to two or more different segments.

This simple and intuitive segmentation principle has some perhaps not so intuitive consequences. As the example (3.1) below shows, one of these consequences is that so-called *inherently reflexive verbs* should be treated as two separate segments: the verbal form itself and the reflexive marker *się*.

(3.1) Bo ja się naprawdę boję głośno roześmiać.
because I Refl really fear loudly lough

'Becuase I'm really afraid to laugh out loudly.'

In the example above, illustrating the so-called haplology of the Polish reflexive marker (Kupść, 1999), a single *się* seems to simultaneously belong to two reflexive verbs: BAĆ SIĘ 'to fear' and ROZEŚMIAĆ SIĘ 'to laugh out'. However, the requirement that segments be disjoint precludes the possibility of (3.1) containing the segments *boję się* and *roześmiać się*, so at least in such cases *się* should be analysed as a separate segment. Since *się* is treated as a separate segment in case of some sentences involving reflexive verbs, it is natural (and supported by *Ockham's razor*) to regard it as a separate segment also in other instances of reflexive verbs.

On the basis of similar reasoning applied to the examples below, also so called *analytic verbal forms*, sequences like *po polsku* 'in Polish', etc., should be split into smaller segments — otherwise, the sequences *będę*, *niech*, *po*, etc., would have to belong to two segments at the same time: *będę szedł* and *będę śpiewał*, *niech przyjdzie* and *niech zaśpiewa*, *po polsku* and *po angielsku*.

(3.2) a. Będę długo szedł i śpiewał.
I will long walk and sing

'I'll be walking and singing for a long time.'

b. Niech no tylko przyjdzie i zaśpiewa!
Let Part only come and sing
'Just let him come and sing!'

(3.3) Mówię po polsku i angielsku.
I speak in Polish and English
'I speak Polish and English.'

On the basis of such cases, a more general principle was adopted in the IPI PAN Corpus, namely, that segments cannot be longer than orthographic *words*, i.e., maximal sequences of characters, excluding word-delimiter characters such as white characters (spaces, tabulation marks, etc.) and punctuation marks (except the hyphen, the apostrophe in forms such *Chomsky'ego* and *(de) l'Hospitala*, and the full stop in abbreviations, initials, etc.). Word-delimiting punctuation marks are regarded to be separate tokens.

Words understood as maximal sequences of non-word-delimiting characters are usually individual segments, although there are cases where — again on the basis of the contiguity and disjointness requirements mentioned above — such words should be split into smaller segments.

(3.4) a. Dawno nie śpiewała*m* i nie tańczyła*m*.
long time ago not sang-I and not danced-I
'I haven't sung and danced for a long time.'

b. Dawno*m* nie śpiewała i nie tańczyła.
long time ago-I not sang and not danced

(3.5) a. Kiedyś zatańczył*by*m i zaśpiewał*by*m tam.
once dance-would-I and sing-would-I there
'Once I would dance and sing there.'

b. Kiedyś *by*m tam zaśpiewał i zatańczył.
once would-I there sing and dance

Example (3.4) shows that the so-called agglutinative forms of the lexeme BYĆ 'to be', i.e., so-called mobile inflections *-(e)m*, *-(e)ś*, *-(e)śmy* and *-(e)ście*, should be treated as separate segments. Similarly, example (3.5) justifies the decision to treat the subjunctive particle *by* as a separate segment. All exceptions from the rule that words, in the sense given above, are single segments are given below.

- Agglutinative forms of the lexeme BYĆ 'to be' are separate segments, so the following words consist of two segments each: |łgał|eś| 'lied-you', |długo|śmy| 'long time-we', |tak|em| 'so-I'.
- Also particles *by* (subjunctive particle), *-ż(e)* (emphatic particle) and *-li* (question particle) are considered to be separate segments, so the following words consist of a number of segments: |przyszedł|by| 'come-would', |napisała|by|m| 'write-would-I', |chodź|że| 'come-Emph', |potrzebował|że|by|ś| 'need-Emph-would-you', |znasz|li| 'know-Q'.
- The post-prepositional weak pronominal form *-ń*, as in |do|ń| 'to-him' or |ze|ń| 'with-him', is also a separate segment.
- Some words containing the hyphen are also split into segments, namely:
  — words such as |polsko|-|niemiecki| 'Polish-German',
  — double names, e.g., |Kowalska|-|Nowakowska|.
  On the other hand, inflected acronyms such as *PRL-u* are not split into smaller segments.
- Sentence-final words containing word-final full stop, e.g., abbreviations such as *itp.* 'etc.', ordinal numbers written in digits, and initials, are also split into smaller segments, e.g.: |itp|.|, |George| |W|.|, etc. The reason for that comes from the double role of the full stop in such cases: it is a part of the word and at the same time it plays the role of a sentence-final punctuation mark (cf. §3.5.1). When such words do not occur in sentence-final positions, they are considered to be single segments.

The segmentation principles given above lead to the segmentation of (3.6) (translated into English in (3.7)) that is presented in (3.8).

(3.6) Pojechalibyśmy z Janem M. Rokitą i Janem Nowakiem-Jeziorańskim na sesję polsko-amerykańską, gdyby nas zaprosił George W. Byłaby to nasza już 2. doń podróż od czasów PRL-u, a może i 3., czy nawet 4.

(3.7) 'We would go with Jan M. Rokita and Jan Nowak-Jeziorański to the Polish-American session, if we were invited by George W. That would already be our 2nd trip to him since the times of PRL, and perhaps 3rd, or even 4th.'

(3.8) *Pojechali by śmy z Janem M. Rokitą i Janem Nowakiem - Jeziorańskim na sesję polsko - amerykańską , gdyby nas zaprosił George W. Była by to nasza już 2. do ń podróż od czasów PRL-u , a może i 3. , czy nawet 4.*

3.2. The structure of morphosyntactic tags

For a given segment, a tag assigned to this segment represents its base form, so-called lemma, as well as a morphosyntactic interpretation of that segment; we will sometimes use the term *tag* in this narrower meaning of morphosyntactic interpretation, which excludes the lemma. In case of punctuation segments, the base form of such a segment is that segment itself, while the tag assigned to such a segment is interp. In what follows we concentrate on the tagset for less trivial (non-punctuation) segments.

Each morphosyntactic tag is a sequence of colon-separated values, e.g.: subst:sg:nom:m1 for the segment *chłopiec* 'boy'. The first value, e.g., subst, determines the *grammatical class* (cf. §3.4), while the values that follow it, e.g., sg, nom and m1, are the values of grammatical categories (cf. §3.3) appropriate for that grammatical class. That means that the tagset adopted in the current project is a *positional tagset*, just like the tagset of the Czech National Corpus or the family of tagsets developed within the Multext-East project (Erjavec, 2001).

3.3. Grammatical categories

The following table presents the repertoire of grammatical categories used in the IPI PAN Corpus.

Number: (2 values)		
singular	sg	*oko*
plural	pl	*oczy*
Case: (7 values)		
nominative	nom	*woda*

genitive	gen	*wody*
dative	dat	*wodzie*
accusative	acc	*wodę*
instrumental	inst	*wodą*
locative	loc	*wodzie*
vocative	voc	*wodo*
Gender: (5 values)		
human masculine (virile)	m1	*papież, kto, wujostwo*
animate masculine	m2	*baranek, walc, babsztyl*
inanimate masculine	m3	*stół*
feminine	f	*stuła*
neuter	n	*dziecko, okno, co, skrzypce, spodnie*
Person: (3 values)		
first	pri	*bredzę, my*
second	sec	*bredzisz, wy*
third	ter	*bredzi, oni*
Degree: (3 values)		
positive	pos	*cudny*
comparative	comp	*cudniejszy*
superlative	sup	*najcudniejszy*
Aspect: (2 values)		
imperfective	imperf	*iść*
perfective	perf	*zajść*
Negation: (2 values)		
affirmative	aff	*pisanie, czytanego*
negative	neg	*niepisanie, nieczytanego*
Accentability: (2 values)		
accented (strong)	akc	*jego, niego, tobie*
non-accented (weak)	nakc	*go, -ń, ci*

Post-prepositionality: (2 values)		
post-prepositional	praep	*niego, -ń*
non-post-prepositional	npraep	*jego, go*
Accommodability: (2 values)		
agreeing	congr	*dwaj, pięcioma*
governing	rec	*dwóch, dwu, pięciorgiem*
Agglutination: (2 values)		
non-agglutinative	nagl	*niósł*
agglutinative	agl	*niosł-*
Vocalicity: (2 values)		
vocalic	wok	*-em*
non-vocalic	nwok	*-m*

The categories of **number**, **case**, **person** and **degree** are understood here in the traditional way and do not require any explanation.

The grammatical category of **gender** is understood as in Mańczak 1956, i.e., unlike in the 'preliminary school grammar', the gender of a noun does not depend on that noun's number. The following contexts might be used to determine the gender of those nouns which have singular forms:[1]

[1] The previous versions of the IPI PAN tagset assumed the repertoire of nine genders proposed in Saloni 1976. Due to some limitations of the current version of the morphological analyser used in the project, as well as because of some doubts (cf. Przepiórkowski *et al.* 2002 and Woliński 2001) about the set of *plurale tantum* genders proposed in Saloni 1976, the current version of the tagset assumes the more conservative repertoire of five genders. Such *plurale tantum* nouns as *wujostwo* 'uncle and aunt', are marked here as m1, while *plurale tantum* forms such as *skrzypce* 'violin', *sanie* 'sleigh', *spodnie* 'trousers' are assumed, partially on the basis of the reasoning presented in Przepiórkowski 2003a, to be neuter forms, marked as n. The following contexts may help determine the gender of *plurale tantum* nouns:

(i) ____ byli ważni. m1

(ii) ____ były ważne. n

(3.9) Widzę jednego ____ z tych, których lubię. m1

(3.10) Widzę jednego ____ z tych, które lubię. m2

(3.11) Widzę jeden ____. m3

(3.12) Widzę jedno ____. n

(3.13) Widzę jedną ____. f

The category of **aspect** is a lexical category: forms do not inflect for aspect, but rather have the value of aspect, constant for all forms of a given verb, determined lexically.

The category of **negation** is appropriate for those verbal forms for which the negative prefix *nie-* is orthographically joined to the verbal form, i.e., this category is useful for distinguishing forms such as *pisanie* 'writing' and *niepisanie* 'non-writing', *napisany* 'written' and *nienapisany* 'unwritten', but not for distinguishing *pisać* 'to write' and *nie pisać* 'not to write'.

The categories of **accentability** and **post-prepositionality** are relevant only for some forms of personal pronouns (in case of post-propositionality — only some forms of 3rd person pronouns).

The category of **accommodability** is appropriate to all numeral forms. The value of this category for a given numeral form is 'agreeing' if and only if that numeral form agrees in case with the accompanying noun. A more detailed discussion of this category can be found in Przepiórkowski 2003b and Woliński 2003.

The two final grammatical categories assumed in the current tagset, **agglutination** and **vocalicity**, are necessary because past forms such as *niosłem* 'carried-I.MASC' and *niosłam* 'carried-I.FEM' are split into segments, as in *niosł*|*em*, *niosła*|*m*. Although in the vast majority of cases the first segment in such forms looks just like the corresponding 3rd person past form, e.g., *ja* *szedł*|*em* 'I walked-I' and *on* *szedł* 'he walked', sometimes these two forms differ, as in *ja* *niosł*|*em* 'I carried-I' and *on* *niósł* 'he carried'. In case of such differences, the form which combines with the agglutinate (e.g., with *-em*), for example the form *niosł-*, will be marked as agglutinative, while the form occurring on its own, e.g., the form *niósł*, will be marked as non-agglutinative. Moreover, the category of vocalicity distinguishes those agglutinates (e.g., *-em*) which attach to forms ending in a consonant, from those (e.g., *-m*) which attach to forms ending in a vowel.

Traditional grammatical categories which are missing in the IPI PAN tagset include tense, mood and voice: such categories are appropriate to units larger than segments.

3.4. Grammatical classes

The basic notion of the present tagset which corresponds to the traditional notion of *part of speech* is *grammatical class*. We will use this term interchangeably with the term *flexemic class*.

The scope of traditional parts of speech such as verb, noun, numeral or pronoun is fuzzy and, hence, controversial. For example, are gerundial forms such as *picie* 'drinking' and *palenie* 'smoking' verbs (they have the category of aspect and they are productively related to verbal forms such as *pić* 'to drink' and *palić* 'to smoke'), or are they nouns (they decline for case, and they have the lexical category of gender)? Are ordinal numerals such as *piąty* 'fifth' numerals (semantically, they are numerals), or are they adjectives (they have adjectival inflection)? Are adjectival pronouns such as *taki* 'such' pronouns (semantics) or adjectives (inflection)?

Grammatical classes used in the IPI PAN Corpus are more precisely delimited and, overall, finer-grained than traditional parts of speech. The classes assumed here are based on the notion of *flexeme*, introduced in Bień 1991, 2004, narrower than the notion of *lexeme*.

3.4.1. Flexemes

Informally speaking, two forms belong to the same lexeme if and only if they mean the same thing (modulo productive differences in meaning resulting from different values of grammatical categories such as number and person), and if they have a similar morphological form,[2] so, for example, the forms *pięć* 'five.NOM', *pięcioma* 'five.INST' and *pięciokrotny* 'fivefold' could be considered to be forms of the same lexeme, just as the forms *wypije* '(s)he will drink up', *wypić* 'to drink up' and *wypito* 'was drunk up' are forms of the same lexeme.

[2] Well-known Polish exceptions to the last requirement include the lexeme ROK 'year', which contains singular forms such as *rokiem* and plural forms such as *latami*, and the lexeme CZŁOWIEK 'man', which contains singular forms such as *człowiekiem* and plural forms such as *ludźmi*.

On the other hand, in case of flexemes, two forms belong to the same flexeme when the requirements above are satisfied, i.e., when they mean the same thing and have similar forms, and — additionally — when they have the same grammatical categories. So, for example, the personal verbal forms *wypije* 'will drink up', *wypijecie* 'you will drink up', *wypijemy* 'we will drink up' all belong to the same flexeme, characterised by the categories of number, gender and aspect, but forms such as *wypić* 'to drink up' or *wypito* 'was drunk up', which do not have the categories of number and gender, will be excluded from that flexeme.

On the basis of the above first approximation of the notion of *flexeme*, the forms such as *wypić* 'to drink up' and *wypito* 'was drunk up' should be classified as belonging to the same flexeme: they have the same grammatical categories, namely, no inflectional (morphological) categories and the sole lexical category of aspect. However, both forms have the same value of that single category (imperfective), so grammatical categories do not distinguish those forms. Such a situation may occur in case of free variants, e.g., *funkcji* and *funkcyj* 'functions.GEN.PL', or *HIT-u* and *HIT-a* 'HIT.GEN.SG', whose syntactic distribution is the same. However, in case of *wypić* and *wypito*, grammatical categories of these forms are the same, but their distribution clearly differs. We will require that forms with identical values of grammatical categories and different syntactic distribution belong to different flexemes, so — in this particular case — we will posit two non-inflecting verbal flexemes, each containing just one form, *wypić* and *wypito*, respectively.

Continuing this line of reasoning, forms of perfective verbs such as WYPIĆ 'to drink up' can be partitioned into the following flexemes:

- so-called l-participle, containing forms which inflect for number and gender, but not for person, e.g., *wypił*, *wypili*, *wypiły*,
- a flexeme containing future forms, inflecting for number and person, but not for gender, e.g., *wypiję*, *wypijemy*, *wypiją*,
- the imperative flexeme, containing forms which also inflect for number and person, but in a defective manner: *wypijmy*, *wypij*, *wypijcie*,
- three non-inflecting flexemes, each containing a single form: infinitive (*wypić*), impersonal form (*wypito*), and anterior adverbial participle (*wypiwszy*),

- gerund, containing forms which inflect for case, negation, and — potentially — for number, and which have lexical gender (always neuter), e.g., *wypicie*, *wypiciem*, *niewypiciu*,
- passive adjectival participle, containing forms which inflect for number, case and gender, e.g., *wypity*, *wypite*, *wypitymi*.

Similarly, the forms of imperfective verbs such as PIĆ 'to drink' could be split into the following flexemes:

- l-participle,
- a flexeme containing present tense forms, e.g.: *piję*, *pijemy*, *pijecie*,
- the imperative flexeme,
- three non-inflecting flexemes: infinitive, impersonal form, and the contemporary adverbial participle (*pijąc*),
- gerund,
- passive adjectival participle,
- active adjectival participle, which also contains forms inflecting for number, case and gender, e.g.: *pijący*, *pijące*, *pijącymi*.

A different set of flexemes should be proposed for the verb BYĆ 'to be': apart from the l-participle (*był*, *byli*, etc.), the present tense flexeme (*jestem*, *jesteśmy*, *jesteście*, etc.), imperative (*bądźmy*, *bądź*, *bądźcie*), infinitive (*być*), contemporary adverbial participle (*będąc*), gerund (*bycie*, etc.), and the active adjectival participle (*będący*, etc.), also:

- a flexeme containing future tense forms, inflecting for number and person, e.g., *będę*, *będziecie*, and
- the agglutinate, i.e., a flexeme containing the forms *-em*, *-śmy*, etc.

Also different forms of adjectives can be partitioned into a small number of flexemes. Most adjectival forms inflect for number, case and gender, and sometimes also for degree. For example, the forms *polski*, *polskiej*, *polskimi*, etc., 'Polish', all belong to the same adjective flexeme. On the other hand, there are two adjectival forms which do not have any of these, or any other, grammatical categories: *polsko*, as in *polsko-niemiecki* 'Polish-German', and *polsku*, as in *po polsku*, lit. 'in Polish'. Since these two non-inflecting forms have different distribution, they will be split into two

different flexemes: ad-adjectival adjective (*polsko*) and post-prepositional adjective (*polsku*).

When delimiting nominal flexemes, we assume that various forms of the same nominal flexeme have the same grammatical gender, so, for example, *fryzjer* 'hairdresser.MASC' and *fryzjerka* 'hairdresser.FEM' are forms of two different flexemes. That means that a typical nominal flexeme, inflecting for case and number, contains 14 forms (for two values of grammatical number and seven values of grammatical case), but there are also *plurale tantum* flexemes, which do not have singular forms, e.g., the flexemes WUJOSTWO 'uncle and aunt', URODZINY 'birthday', and SPODNIE 'trousers', as well as *singulare tantum* nouns, which do not have plural forms, e.g., the flexemes KTO 'who' and CO 'what'.

In case of human masculine nouns, it is not clear how to treat the so-called depreciative forms, e.g., *profesory* as in *Przyszły głupie profesory i naniosły błota* 'Stupid professors came and brought in some mud', lit. 'Came stupid professors and brought in mud', as opposed to the ordinary non-depreciative form *profesorowie*, as in *Przyszli głupi profesorowie i nanieśli błota*. How should non-depreciative forms such as *profesorowie* be distinguished from depreciative forms such as *profesory*? One possibility is to introduce a new grammatical category, let us call it *depreciation*, which would differentiate between depreciative and non-depreciative forms. However, this solution would lead to complications at the syntactic level, or more precisely, at the level at which agreement between such hypothetical human masculine depreciative forms (*profesory*) and the non-human masculine adjectival (*głupie*) and verbal (*przyszły*, *naniosły*) forms with which they co-occur is described. For that reason, a different solution was adopted in the current tagset, namely, a solution which consists in analysing such depreciative forms as belonging to a separate class of depreciative flexemes. Such depreciative flexemes contain only two forms, both of the masculine animate gender, in this case: *profesory*, which differ only in the value of case, i.e., which are in the nominative or in the vocative case.

According to our understanding of flexemes, also numeral forms should be partitioned into a number of flexemes:

- main numeral contains forms such *pięć* 'five.NOM/ACC', *pięciu* 'five.ACC/ /DAT/LOC' and *pięcioma* 'five.INST'; such forms inflect for case, gender and — in a defective manner — for accommodability, but they have lexically determined number (usually plural),

- collective numeral contains forms such as *pięcioro* 'five.NOM/ACC' and *pięciorgiem* 'five.INST'; they have lexically determined plural number and neuter gender, and they inflect for case and accommodability (in a defective manner),
- forms such as *piąty* 'fifth.SG.NOM.MASC', *piąta* 'fifth.SG.NOM.FEM', *piątymi* 'fifth.PL.INST', etc., constitute an adjectival flexeme;
- also forms such as *pięciokrotny* 'fivefold.SG.NOM.MASC', *pięciokrotnemu* 'fivefold.SG.DAT.MASC', etc., constitute a separate adjectival flexeme.

Most of the traditional pronominal forms belong to adjectival flexemes (e.g., *taki* 'such', *jakiś* 'some', *który* 'which', etc.), nominal flexemes (e.g., *kto* 'who', *coś* 'something'), etc. However, because of its idiosyncratic inflection, it is convenient to distinguish a separate flexeme for the reflexive pronoun SIEBIE, whose forms seem to inflect only for case, as well as separate flexemes for personal pronouns, which have a rather complex and idiosyncratic paradigm.

Other inflecting flexemes include gradable adverbs, as well as flexemes containing forms such as *winien* 'ought.SG.MASC.3RD', *winna* 'ought.SG.FEM .3RD', *winniśmy* 'ought.PL.1ST', etc. Other flexemes are non-inflecting flexemes, i.e., they contain single forms, e.g., the flexeme ORAZ, containing the conjunction *oraz*, or the flexeme NA, containing the preposition *na*.

3.4.2. Flexemic classes

Just as flexemes are non-empty and disjoint sets of those word forms which have uniform semantic, morphological, morphosyntactic, and — to a certain extent — distributional properties, *flexemic classes* are non-empty, disjoint, morphosyntactically and — to a certain extent — distributionally uniform sets of flexemes.

The following table contains the rough morphosyntactic characteristics of all flexemic classes assumed in the present tagset. The symbol $\oplus$ in the table means that, for a given flexemic class, a given grammatical category is a morphological category (flexemes belonging to this class normally inflect for that category), while the symbol $\odot$ means that the category is a lexical category (for each flexeme belonging to this class, all forms of that flexeme have the same value of that category, although that value may differ between flexemes, as in the case of the gender of nouns).

	number	case	gender	person	degree	aspect	negation	accentability	post-prepositionality	accommodability	agglutination	vocalicity
noun	⊕	⊕	⊙									
depreciative form	⊙	⊕	⊙									
main numeral	⊙	⊕	⊕							⊕		
collective numeral	⊙	⊕	⊙							⊕		
adjective	⊕	⊕	⊕		⊕							
ad-adj. adjective												
post-prep. adjective												
adverb					⊕							
pronoun (non-3rd person)	⊙	⊕	⊕	⊙				⊕				
pronoun (3rd person)	⊕	⊕	⊕	⊙				⊕	⊕			
pronoun SIEBIE		⊕										
non-past form	⊕			⊕		⊙						
future BYĆ	⊕			⊕		⊙						
agglut. BYĆ	⊕			⊕		⊙						⊕
l-participle	⊕		⊕			⊙					⊕	
imperative form	⊕			⊕		⊙						
impersonal form						⊙						
infinitive						⊙						
adv. contemp. prtcp.						⊙						
adv. anter. prtcp.						⊙						
gerund	⊕	⊕	⊙			⊙	⊕					
adj. act. prtcp.	⊕	⊕	⊕			⊙	⊕					
adj. pass. prtcp.	⊕	⊕	⊕			⊙	⊕					
winien-like verb	⊕		⊕			⊙						
predicative												
preposition		⊙										
conjunction												
particle-adverb												
alien (nominal)	⊕	⊕	⊙									
alien (other)												
unknown form												
punctuation												

A more detailed characterisation of the morphosyntactic and, in some cases, distributional features of particular flexemic classes is given below.

noun contains flexemes inflecting for number and case, with a lexically determined grammatical gender, which do not have the category of person, e.g., WODA 'water', PROFESOR 'professor', PIĘCIOKROTNOŚĆ 'five-foldness'; this class also contains defective *plurale tantum* and *singulare tantum* flexemes, but not depreciative flexemes,

depreciative form contains depreciative flexemes, i.e., flexemes with fixed number (plural) and gender (animate masculine), defectively inflecting for case (only nominative and vocative), e.g., PROFESORY 'professors', STUDENTY 'students',

main numeral contains flexemes inflecting for case, gender and — defectively — for accommodability, with lexically determined number (normally plural), i.e., flexemes such as PIĘĆ 'five' and WIELE 'many', also including defective numeral flexemes such as TROCHĘ 'some' and DUŻO 'much, many', whose case values are limited to nominative, accusative and genitive,

collective numeral contains flexemes which inflect for case and — defectively — for accommodability, with lexically determined number (always plural) and gender (always neuter), i.e., flexemes such as PIĘCIORO 'five',

adjective contains flexemes inflecting for number, case and gender, as well as — in some cases — for degree, e.g., MIŁY 'nice', TECHNICZNY 'technical', TAKI 'such', KTÓRY 'which', PIĄTY 'fifth', WIELOKROTNY 'manifold' and JEDEN 'one',

ad-adjectival adjective contains non-inflecting de-adjectival flexemes such as POLSKO 'Polish' and NIEMIECKO 'German',

post-prepositional adjective contains non-inflecting de-adjectival flexemes such as POLSKU 'Polish' and NIEMIECKU 'German',

adverb contains those flexemes which only inflect for degree (gradable adverbs, e.g., BARDZO 'very', MIŁO 'nicely'), as well as non-inflecting de-adjectival flexemes which do not belong to the two previous classes (non-gradable de-adjectival adverbs, e.g., TECHNICZNIE),

non-3rd person pronoun contains exactly four flexemes, which inflect for case and gender, but have lexically determined number and person: JA 'I', MY 'we', TY 'you.SG', WY 'you.PL'; some forms of the flexemes JA and TY additionally inflect for accentability (e.g., *ci* vs. *tobie*, 'you.SG.DAT'),

3rd person pronoun contains exactly one flexeme, ON 'he', with lexically determined person (3rd), inflecting for number, case and gender; some forms additionally inflect for accentability and post-prepositionality (e.g., *niego* vs. *go*, 'him.ACC'),

siebie also contains exactly one flexeme, SIEBIE, apparently the only Polish flexeme which inflects only for case (defectively, without nominative and vocative forms),

non-past form contains flexemes which inflect for number and person, and have the lexical category of aspect: future forms (with perfective aspect), e.g., *wypiję* 'I will drink up', and present forms (with imperfective aspect), e.g., *piję* 'I am drinking',

future form of BYĆ contains just one flexeme, consisting of the future forms of the imperfective verb BYĆ 'to be': *będę*, *będziesz*, etc.

agglutinate BYĆ contains one flexeme, consisting of the agglutinative forms of BYĆ: *-m*, *-em*, *-śmy*, etc.,

l-participle contains flexemes inflecting for number and gender, with a lexically determined value of aspect, e.g., the flexeme containing the forms *niósł*, *niosł-*, *niosła*, *nieśli*, *niosły* 'carry',

imperative also contains flexemes with a lexically determined value of aspect, inflecting for number and gender, but only defectively so (only the 1.SG, 2.SG and 2.PL forms), e.g., the flexeme containing the forms *pij* 'you.SG drink!', *pijcie* 'you.PL drink', *pijmy* 'let us drink!',

impersonal consists of single-element flexemes containing non-inflecting aspectual forms ending in *-no* or *-to*, e.g., PITO 'was drunk',

infinitive consists of single-element flexemes which contain infinitive forms, e.g., PIĆ 'to drink',

contemporary adverbial participle consists of single-element flexemes containing imperfective adverbial participles, e.g., PIJĄC 'drinking',

anterior adverbial participle consists of single-element flexemes containing perfective adverbial participles, e.g., WYPIWSZY 'having drunk up',

gerund contains flexemes which inflect for number, case and negation, and have the lexical categories of gender (always neuter) and aspect, e.g., PICIE 'drinking' and WYPICIE 'drinking up',

active adjectival participle contains active adjectival participles, inflecting for number, case, gender and negation, with lexical aspect (always imperfective), e.g., PIJĄCY 'drinking',

passive adjectival participle contains passive adjectival participles, inflecting for number, case, gender and negation, with lexical aspect, e.g., PITY 'drunk', WYPITY 'drunk up',

winien contains the flexemes WINIEN 'should', POWINIEN 'should' and RAD 'eager, pleased', inflecting for number and gender, with only analytical past tense and conditional forms,

predicative contains non-inflecting flexemes such as BRAK 'to lack, to miss', TRZEBA (deontic modality), WARTO 'to be worth it', etc., which analytically inflect for tense and mood (e.g., *było warto*, *warto*, *warto by*, *będzie warto*),

preposition contains non-inflecting prepositional flexemes, which have the lexical category of case, indicating the subcategorisation properties of the preposition,[3] and which do not occur with non-post-prepositional forms of pronouns: BEZ, BEZE, CO, DLA, DO, DOKOŁA, DOOKOŁA, DZIĘKI, KOŁO, KONTRA, KU, MIĘDZY, MIMO, NA, NAD, NADE, NAOKOŁO, NAPRZECIW, NAPRZECIWKO, O, OBOK, OD, ODE, ODNOŚNIE, OPRÓCZ, PER, PO, POD, PODE, PODCZAS, PODLE, PODŁUG, POMIĘDZY, POMIMO, PONAD, PONIŻEJ, POPRZEZ, POŚRODKU, POŚRÓD, POWYŻEJ, POZA, PRÓCZ, PRZECIW, PRZECIWKO, PRZED, PRZEDE, PRZEZ, PRZEZE, PRZY, SPOD, SPODE, SPOMIĘDZY, SPOŚRÓD, SPOZA, SPRZED, ŚRÓD, U, W, WE, WBREW, WEDLE, WEDŁUG, WEWNĄTRZ, WOBEC, WOKOŁO, WOKÓŁ, WSKUTEK, WŚRÓD, WZDŁUŻ, Z, ZA, ZE, ZAMIAST, ZEWNĄTRZ, ZNAD, ZNADE, ZZA,

conjunction contains non-inflecting coordinating flexemes: A, ABY, ACZKOLWIEK, ALBO, ALBOWIEM, ALE, ALEŻ, ANI, ANIŻELI, AŻ, AŻEBY, BĄDŹ, BO, BOWIEM, BY, BYLE, CHOCIAŻ, CHOCIAŻBY, CHOĆ, CHOĆBY, CZY, CZYLI, DOPÓKI, DOPÓTY, GDY, GDYBY, GDYŻ, I, IM, IŻ, IŻBY, JAK, JAKBY, JAKKOLWIEK, JAKO, JAKOBY, JEDNAK, JEDNAKŻE, JEŚLI, JEŚLIBY, JEŻELI, KIEDY, LECZ, LUB, NATOMIAST, NI, NIM, NIŻ,

[3] It should be noted that the meaning of the category of case for prepositions differs substantially from the meaning of the same category for other grammatical classes.

ORAZ, PONIEWAŻ, PÓKI, PÓTY, PRZETO, SKORO, TAK, TEDY, TO, TOTEŻ, TYLKO, TYM, WIĘC, ZAMIAST, ZANIM, ZARÓWNO, ZAŚ, ZATEM, ŻE, ŻEBY,

particle-adverb contains non-inflecting flexemes which do not fit any of the previous classes, e.g., JUŻ 'already', ZBYT 'too', SIĘ (reflexive marker), NIE (negation marker), BY (subjunctive particle), -LI (question particle), OJ (interjection), AHA (interjection), etc.

Apart from the classes listed above, the present tagset introduces additional four classes:

nominal alien contains foreign expressions, mathematical and chemical formulae, etc., which occupy a nominal position in the sentence and, hence, may be assigned the values of number, case and gender,

other alien foreign expressions, mathematical and chemical formulae, etc., which do not occupy a nominal position and, hence, are treated here as non-inflecting,

unknown form contains forms which have not been recognised in the process of morphological analysis,

punctuation contains non-inflecting punctuation 'flexemes', e.g., :, ., !, etc.

3.4.3. Lemmata

As mentioned above, complete morphosyntactic tags assigned to a particular segment contain not only the interpretation of this segment in terms of grammatical classes and categories, but also the base form of that segment in those interpretations, or its lemma. But what should the base form of, e.g., the segment *idziemy* 'walk.1PL, go.1PL' be? Should it be one of the forms belonging to the same flexeme as *idziemy*, e.g., *idę* 'walk.1SG, go.1SG', or should it be the traditional base form, i.e., the infinitive form *iść*, even though it belongs to a different flexeme?

The stance adopted in the IPI PAN Corpus follows tradition: segments are assigned traditional base forms such as infinitive or single masculine nominative forms, even if such a base form does not belong to the same flexeme as the segment itself.

The following table provides the information about base forms for all grammatical classes, as well as the abbreviations of these classes as used in the IPI PAN Corpus.

flexeme	abbreviation	base form	example
noun	subst	singular nominative	*profesor*
depreciative form	depr	singular nominative form of the corresponding noun	*profesor*
main numeral	num	inanimate masculine nominative form	*pięć, dwa*
collective numeral	numcol	inanimate masculine nominative form of the main numeral	*pięć, dwa*
adjective	adj	singular nominative masculine positive form	*polski*
ad-adjectival adjective	adja	singular nominative masculine positive form of the adjective	*polski*
post-prepositional adjective	adjp	singular nominative masculine positive form of the adjective	*polski*
adverb	adv	positive form	*dobrze, bardzo*
non-3rd person pronoun	ppron12	singular nominative	*ja*
3rd-person pronoun	ppron3	singular nominative	*on*
pronoun SIEBIE	siebie	accusative	*siebie*
non-past form	fin	infinitive	*czytać*
future BYĆ	bedzie	infinitive	*być*
agglutinate BYĆ	aglt	infinitive	*być*
l-participle	praet	infinitive	*czytać*
imperative	impt	infinitive	*czytać*
impersonal	imps	infinitive	*czytać*
infinitive	inf	infinitive	*czytać*
contemporary adv. participle	pcon	infinitive	*czytać*
anterior adv. participle	pant	infinitive	*czytać*
gerund	ger	infinitive	*czytać*

active adj. participle	pact	infinitive	*czytać*
passive adj. participle	ppas	infinitive	*czytać*
winien	winien	singular masculine form	*powinien, rad*
predicative	pred	the only form of that flexeme	*warto*
preposition	prep	the only form of that flexeme	*na, przez, w*
conjunction	conj	the only form of that flexeme	*oraz*
particle-adverb	qub	the only form of that flexeme	*nie, -że, się*
nominal alien	xxs	singular nominative form	*de, l'Hospital*
other alien	xxx	the only form of that flexeme	*bene*
unknown form	ign	the only form of that flexeme	
punctuation	interp	the only form of that flexeme	*;, ., (,]*

3.5. Idiosyncratic segments of written Polish

The morphosyntactic annotation of written Polish texts requires making a number of decisions about the segmentation and tagging of sequences of characters which lie at the border of the interest of linguists and typographers. This section discusses several classes of such sequences typical for written texts.

3.5.1. Haplology of the full stop

Some natural language forms end in the full stop, e.g.:

- abbreviations like *np.* 'e.g.', *itp.* 'etc.', and, in some case positions, *dr.* 'Dr.', *mgr.* 'M.Sc.', etc.,
- ordinal numbers written in digits,
- initials.

It is not *a priori* clear how the full stop should be treated in such forms, when they occur in sentence-final positions, i.e., when the full stop also marks the end of the sentence, e.g.:

(3.14) Działo się to w 1945 r.
happened Refl this in 1945 yr.
'This was happening in 1945.'

(3.15) Czy to 3. pacjent? Nie, 2.
Q this 3rd patient no 2nd
'Is this the third patient? No, it's the second.'

(3.16) Prezydenta Stanów Zjednoczonych zwą George W.
President.ACC States.GEN United.GEN call.3PL George.NOM W.NOM
'The president of the United States is called George W.'

The solution adopted in the IPI PAN tagset is to always treat the full stop at the end of a sentence as a punctuation mark, even if it is a part of an abbreviation, an ordinal number or an initial. So, in the examples above, the full stop is a separate segment: *r*|.|, 2|.|, *W*|.|.

On the other hand, in case the full stop in such forms does not play a double role of sentence-final punctuation, it is not considered to be a separate segment. For example, the forms *r.*, *2.* and *W.* are single segments in sentences below.

(3.17) Działo się to w 1945 r.!
happened Refl this in 1945 yr.
'This was happening in 1945!'

(3.18) Czy to 3. pacjent? Nie, to 2. pacjent
Q this 3rd patient no this 2nd patient
'Is this the third patient? No, it's the second patient.'

(3.19) Obecny prezydent Stanów Zjednoczonych nazywa się
current.NOM president.NOM States.GEN United.GEN called.1SG Refl
George W. Bush.
George W. Bush
'The current president of the United States's name is George W. Bush.'

The base forms of abbreviations are also abbreviations, written without the full stop in case of abbreviations such as *wg*, *dr* and *mgr*, and with the full stop in case of abbreviations such as *hab.*, *itp.*, or *np.*, following Polish orthography. In case of ordinal numbers, the base form is the same number, ending in the full stop (even if it occurred without the full stop in the text). Moreover, base forms of initials are the same initials, also always spelled with the full stop, e.g.:

(3.20) Klawiatura, myszka itp. są wliczone w cenę komputera.
keyboard mouse etc. are included in price computer
'Keyboard, mouse, etc., are included in the price of the computer.'

- segment: *itp.*
- base form: *itp.*

(3.21) Wliczone w cenę komputera są klawiatura, myszka itp.
included in price computer are keyboard mouse etc.
'Keyboard, mouse, etc., are included in the price of the computer.'

- segment: *itp.*
- base form: *itp.*

(3.22) Działo się to w 1945 r.!
happened Refl this in 1945 yr.
'This was happening in 1945!'

- segments: *1945*, *r.*
- base forms: *1945.*, *r.*

(3.23) Działo się to w 1945 r.
happened Refl this in 1945 yr.
'This was happening in 1945.'

- segments: *1945*, *r*
- base forms: *1945.*, *r.*

(3.24) To 3. pacjent.
this 3rd patient
'This is the third patient.'

- segment: *3.*
- base form: *3.*

(3.25) Nie, to już 4.
no this already 4th

'No, it's already the fourth.'

- segment: *4*
- base form: *4.*

(3.26) To George W. Bush.
this George W. Bush.

'This is George W. Bush.'

- segment: *W.*
- base form: *W.*

(3.27) Ale zwą go George W.
but call him George W.

'But they call him George W.'

- segment: *W*
- base form: *W.*

(3.28) Oto mgr Kwaśniewski.
this M.Sc.NOM Kwaśniewski.NOM

'This is Kwaśniewski, M.Sc.'

- segment: *mgr*
- base form: *mgr*

(3.29) Rozmawiałem z mgr. Kwaśniewskim.
talked with M.Sc.INST Kwaśniewski.INST

'I talked to Kwaśniewski, M.Sc.'

- segment: *mgr.*
- base form: *mgr*

3.5.2. Abbreviations

Segmentation and lemmatisation of abbreviations is discussed in §3.5.1 above. Morphosyntactic interpretations of abbreviations of single segments should correspond to the interpretations of the full forms of those segments. For example, the abbreviation *mgr.* (for *magister* 'M.Sc.') in *Rozmawiałem z mgr. Kwaśniewskim* 'I talked to Kwaśniewski, M.Sc.' should be the tag subst:sg:inst:m1, just as it would be for the full form *magistrem* in this context.

In case of abbreviations of multi-word expressions, the flexemic class of such abbreviations is determined on the basis of their inflection and distribution, e.g.:

- particle-adverbs: *itp.* 'etc.', *itd.* 'etc.', *np.* 'e.g.', *etc.* 'etc.', *jw.* 'as above',
- adjectives: *tzw.* 'so-called', *śp.* 'R.I.P.',[4] *ww.* 'mentioned above',
- nouns: *br.* 'current year', *cd.* 'contd.', lit. 'continuation',
- prepositions: *ds.* 'responsible for' (prep:gen), *pt.* 'under the title of' (prep:nom),
- verbal non-past forms: *cdn.* 'will be continued'.

3.5.3. Numbers

The base form of a number spelled in digits is the same number, with the full stop in case of ordinal numbers (cf. §3.5.1). In case that number is interpreted as ordinal, it is tagged as an adjective, just as ordinal numerals are; otherwise it receives the main numeral interpretation, with the exception of the number *1*, which is interpreted as an adjective, just like the form *jednego* 'one', e.g.: *jednego*, np.:

(3.30) Dałem to *21.* pacjentowi. adj:sg:dat:m1:pos
gave this 21st patient
'I gave this to the 21st patient.'

(3.31) Dałem to *21* pacjentom. num:pl:dat:m1:congr
gave this 21 patients
'I gave this to 21 patients.'

[4] Classified as adjective on the basis of its distribution.

(3.32) Dałem to *1.* pacjentowi. adj:sg:dat:m1:pos
gave this 1st patient
'I gave this to the 1st patient.'

(3.33) Dałem to *1* pacjentowi. adj:sg:dat:m1:pos
gave this 1 patient
'I gave this to one patient.'

(3.34) *0* komputerów zostało sprzedanych. num:pl:nom:m3:rec
0 computers got sold
'0 computers were sold.'

Negative integers are interpreted just as positive integers, with the only exception of *-1*, interpreted as a numeral, not as an adjective (unless it has an ordinal interpretation). The initial minus sign is a part of such numeral segments. Other real numbers are tagged as numerals.

3.5.4. Names and initials

First names, surnames and initials are tagged as nouns, even if they have apparently adjectival declension. The gender of such nouns depends on the natural gender of the person bearing the name, i.e., it is either m1 or f. The segmentation and lemmatisation of initials is discussed in §3.5.1.

3.5.5. Special symbols: %, $, €, ¥, etc.

Base forms of symbols such as %, $, € and ¥ are the same symbols, while their tags are the same as the tags of the corresponding full forms:

(3.35) Kosztowało to 5$.
cost this 5$
'This cost 5$.'

- base form: *$*
- tag: subst:pl:gen:m3

(3.36) Już tylko 5% wyborców nie popiera Leppera.
already only 5% voters not supports Lepper
'Only 5% voters do not support Lepper now.'

- base form: %
- tag: subst:sg:nom:m3

4

Corpus search

The binary version of the IPI PAN Corpus enclosed on the CD-ROM should be accessed via Poliqarp,[1] a search and concordance engine created within the present project by Zygmunt Krynicki and Daniel Janus, under the supervision of the author.[2] Poliqarp has the ambition to be a universal search engine and concordancer: it reads the external specification of the tagset and it uses the universal character encoding scheme, UTF-8. This means that it should be possible to use Poliqarp also with corpora other than the IPI PAN Corpus, including corpora of languages other than Polish.

[1] *POLyinterpretation Indexing Query and Retrieval Processor*.
[2] Mateusz Przepiórkowski took part in the initial design and development phase.

There are three versions of Poliqarp:

- the graphical version, described in §4.2.2, for the following operating systems: Windows 2000, Windows XP (and possibly other Windows systems, but the program was not tested for those) and GNU/Linux;
- text version, described in §4.2.3, for GNU/Linux;
- WWW version, described in §4.2.1, accessible with any Internet browser such as Mozilla, Internet Explorer, Opera or Links.

What is common for these three versions is the rich query syntax, described in §4.1.

4.1. Query syntax

Poliqarp's query syntax is based on that of Corpus Query Processor (CQP), perhaps the most popular program of this kind, created at the University of Stuttgart (Christ, 1994), but it contains a number of additional features and improvements.[3] The present section describes the syntax of Poliqarp queries and illustrates it with numerous examples.

4.1.1. Searching for orthographic forms

In the simplest case, a query is just a sequence of segments, e.g.:

(4.1) `przyszedł czas`

(4.2) `przyszedł em rano`

There are three segments in query (4.2) above, corresponding to two words (cf. §3.1): *przyszedłem* and *rano*. In the case of simple queries like the two queries above, Poliqarp attempts to identify those words which might consist of smaller segments and to handle them properly, so also the following queries will give the expected results:

(4.3) `przyszedłem rano`

(4.4) `długom szedł`

[3] Although the query syntax of Poliqarp is based on that of CQP, Poliqarp was implemented from scratch within the current project and it does not contain any CQP code.

In case of the latter query, Poliqarp will find all occurrences of the three-segment sequence [*długo*][*m*] [*szedł*], interpretable as an adverb (*długo* 'long'), an agglutinate (*-m* 'be'), and an l-participle (*szedł* 'walk, go'), as well as all occurrences of the two-segment sequence [*długom*] [*szedł*], where the first segment is interpreted as a dative nominal form (*długom* 'debts'), and the second — again, as an l-participle.

By default, queries are interpreted in a case-sensitive manner, so the following queries will produce different results:

(4.5) `przyszedł`

(4.6) `Przyszedł`

In order to find all occurrences of the form *przyszedł*, regardless of case, the flag `/i` should be used. Thus, the two queries below will produce the same results, which will in particular contain all results of both queries above.

(4.7) `przyszedł/i`

(4.8) `Przyszedł/i`

Both in the graphical version and in the text version of Poliqarp, case sensitivity can be set globally, for a whole query or a series of queries, cf. §§4.2.2 and 4.2.3.

Queries may contain standard regular expressions over characters, specified with the help of the following special characters: `?`, `*`, `+`, `.`, `,`, `|`, `{`, `}`, `[`, `]`, `(`, `)`, as well as natural numbers; segment specifications containing regular expressions must be enclosed in quotes `"`. Since the formal introduction of regular expressions lies far outside the scope of the current publication, we will be content with discussing just a few examples, which, nevertheless, should allow the user to understand the syntax and semantics of such regular expressions.

(4.9) `"Ala|Ela"`
the character `|` introduces the alternative of two expressions, so the query above can be used to find all occurrences of segments of the form *Ala* or *Ela*,

(4.10) `"[AE]la`
square brackets denote the alternative of characters within them,

so the query above can be used to find those segments whose first character is *A* or *E*, and the following two characters are *la*, i.e., this query is equivalent to the previous query,

(4.11) `"beza?"`
the question mark signals the optionality of the character or the expression in parentheses which immediately precedes it, so the question above will be used find all occurrences of the segments *bez* and *beza*,

(4.12) `"bez."`
the period denotes any character, so the results of this query will include *beza*, *bezy*, *bezą*, etc., but not *bez* or *bezami*,

(4.13) `"bez.?"`
bez, *beza*, *bezy*, *bezą*, etc., but not *bezami*,

(4.14) `".z.z."`
5-character segments, where 2nd and 4th characters are *z* (e.g., *czczą* and *rzezi*),

(4.15) `".z.z..?"`
segments of length 5 or 6, where 2nd and 4th characters are *z* (e.g., *czczą*, *rzezi* and *szczyt*),

(4.16) `"a*by"`
the asterisk denotes any number of occurrences of the character or the expression in parentheses which immediately precedes it, so this query can be used to find segments beginning with any number of *a*s, followed by *by*, e.g., *by* (zero occurrences of *a*), *aby*, *aaaaby*, etc.,

(4.17) `"Ala.*"`
segments beginning with *Ala*, e.g., *Ala* and *Alabama*,

(4.18) `"ala.*"/i`
segments beginning with *ala*, *Ala*, *aLa*, *ALA*, etc., e.g., *Ala*, *alabaster* and *ALABAMA*,

(4.19) `".*al+"`
the plus has a similar interpretation as the asterisk: it denotes any number *greater than zero* of occurrences of the character or the expression in parentheses which immediately precedes it, so this query can be used to find segments ending in *al*, *all*, *alll* etc., but not in *a*, e.g., *dal*, *robal* and *Gall*,

(4.20) `"a{1,3}b.*"/i`
the expression of the form `{n,m}` denotes from `n` to `m` occurrences of the character or the expression in parentheses which immediately precedes it; in this case, the query above can be used to find segments beginning with 1 to 3 occurrences of *a* or *A*, followed by *b* or *B*, and then followed by any sequence of characters, e.g., *Aby*, *aaaby*, *absolutnie*, *ABBA*,

(4.21) `".*(la){3,}.*"`
`{n,}` means at least `n` occurrences, so this query will help to find segments which contain at least three occurrences of the sequence *la* in a row, e.g., *tralalala*, *sialalala*,

(4.22) `"[bcćdfghjklłmnńprsśtwzźż]{4,}[aąeęioóuy]"/i`
segments consisting of at least 4 consonants and exactly 1 vowel, e.g., *źdźbła* i *Chrzczę*,

(4.23) `"([bcćdfghjklłmnńprsśtwzźż]{3}[aąeęioóuy]){2,}"/i`
segments consisting of at least two sequences of the type CCCV, where C is a consonant, and V is a vowel, e.g., *wszystko*, *Zdmuchnąwszy* i *Szmajdziński*; `{n}` means exactly `n` occurrences,

(4.24) `"(pod|na|za)jecha.*"`
segments beginning with *podjecha*, *najecha* or *zajecha*, e.g., *podjechał*, *zajechawszy*.

The specifications of segments given above must match complete segments, rather than only their parts, hence the necessity of flanking the sequence `(la){3,}` in query (4.21) above with the regular expression `.*`, matching any sequence of characters (also the empty sequence). The same effect can be achieved with the help of the flag `/x`, which means that the given specification must be matched by a subsequence of the segment, not necessarily by the complete segment:

(4.25) `"(la){3,}"/x`
segments which contain at least three occurrences of the sequence *la* in a row, e.g., *tralalala*, *sialalala*,

(4.26) `"(la){3,}"/ix`
segments which contain a sequence like *lalala*, *LaLAla*, etc., e.g., *tralalala*, *SiaLaLALA*.

4.1.2. Searching for base forms

The following query may be used in order to find all forms of the lexeme KORPUS:

(4.27) `[base=korpus]`

The `base` attribute is one of many attributes that may be used in a query. The value of this attribute should specify the base form (the lemma) in the sense of §3.4.3, so a query like `[base=pisać]` can be used to find forms such as *pisać* 'write' (infinitive), *piszę* (non-past form), *pisała* (l-participle), *piszcie* (imperative), *pisanie* (gerund), *pisano* (impersonal), *pisane* (adjectival participle), etc.

Another attribute that may be used in queries is `orth`. The values of this attribute specify segments, so each of the following pairs contains queries which are equivalent.

(4.28) a. `przyszedł`

b. `[orth=przyszedł]`

(4.29) a. `Przyszedł/i`

b. `[orth=Przyszedł/i]`

(4.30) a. `przyszedł czas`

b. `[orth=przyszedł][orth=czas]`

On the other hand, the two queries below are *not* equivalent:

(4.31) a. `przyszedłem rano`

b. `[orth=przyszedłem][orth=rano]`

In the first case, Poliqarp will guess that the word *przyszedłem* may consist of two segments, *przyszedł* and *em*, and will expand the query accordingly, as described in §4.1.1. In contrast, the value of `orth` is always interpreted as the specification of a single segment.

The values of `base` and `orth` may contain regular expressions of the kind described in §4.1.1 above, e.g.:

(4.32) `[orth="bez.?"/i]`
find segments *bez*, *Beza*, *bezy*, etc., but not *bzem* or *bezami*,

(4.33) `[base="bez.?"/i]`
find all segments whose base form has 3 or 4 characters and starts with *bez* (understood in a case-insensitive manner), e.g., the segments *bzem*, *bez*, *bezami*, etc.

4.1.3. Higher order queries

Queries about segments and about base forms may be combined. For example, the following query may be used to find all occurrences of the segment *minę* understood as a form of the lexeme MINA 'mine, face' (and not, say, as a form of the lexeme MIJAĆ, 'to pass'):

(4.34) `[orth=minę & base=mina]`

A similar effect can be achieved with the help of the following query, about those occurrences of the segment *minę* which are *not* interpreted as forms of MIJAĆ.

(4.35) `[orth=minę & base!=mijać]`

The condition that the base form be different from *mijać* may also be specified by putting the negation (the exclamation mark) before the name of the attribute, so the query below is equivalent to the query above.

(4.36) `[orth=minę & !base=mijać]`

Just as in the propositional calculus, double negation is equivalent to no negation, so the following queries about the segment *nie* understood as a form of the pronoun ON are fully equivalent:

(4.37) `[orth=nie & base=on]`

(4.38) `[orth=nie & !base!=on]`

(4.39) `[orth=nie & !!!base!=on]`

(4.40) `[orth=nie & !!base=on]`

In Poliqarp queries, the operator `&` plays the role of logical conjunction. The operator dual to `&` is `|`, which plays the role of logical disjunction, e.g.:

(4.41) `[base=on | base=ja]`
find all forms of lexemes ON and JA, equivalent to `[base="on|ja"]`,

(4.42) `[base=on | orth=mnie | orth=ciebie]`
find all forms of the lexeme ON, as well as the segments *mnie* and *ciebie*,

(4.43) `[orth=pora & !(base=por | base=pora)]`
find those occurrences of the segment *pora* which are not interpreted as forms of the lexemes POR or PORA.

In order to better understand the difference between the operators `&` and `|`, let us compare the effect of the following two queries:

(4.34) `[orth=minę & base=mina]`

(4.44) `[orth=minę | base=mina]`

The result of the former query will consist of those segments which simultaneously (conjunction) have the orthographic form *minę* and are interpreted as a form of the lexeme MINA. On the other hand, the result of the latter query will consists of segments which either (disjunction) have the orthographic form *minę*, regardless of the interpretation of this segment, or are a form of the lexeme MINA (e.g., *mina*, *miny*, *minami*). Hence, the latter query should return many more results than the former query.

As the examples above show, specifications of corpus positions, enclosed in square brackets, may contain any number of conditions of the type `attribute=value`, combined with the operators `!`, `&` and `|`. It is also possible to completely omit any conditions — the query below could be used to find all segments in the corpus.[4]

(4.45) `[]`

This trivial specification of corpus positions, matching any segment, may be useful for finding two forms in a certain distance from each other, e.g., two segments separated by two other segments, as in the following query:

(4.46) `[orth=się][][][base=bać]`

The result of this query will include sequences such *się nikogo nie bać*, *się Boga nie boicie*, etc.

[4] Could be, if not for the fact that Poliqarp contains various internal constraints on the number of results of a query.

It would perhaps be more interesting to specify the *upper* limit on the number of segments which may intervene between two forms, not just the exact number of such intervening positions. Poliqarp makes it possible to pose such queries, as it allows to posit regular expressions also over corpus positions. For example, the following query may be used to find a form of the lexeme BAĆ occurring two, three or four positions after the segment *się*:

(4.47) `[orth=się][]{2,4}[base=bać]`

The result of this query will contain all the sequences found by the previous query, as well as sequences such as *się każdy następny Rywin będzie bał*.

A more accurate query concerning various occurrences of the inherently reflexive verb BAĆ SIĘ should find *się* within a certain window before a form of the lexeme BAĆ, but without any intervening punctuation (intervening punctuation will often indicate clause boundary), or immediately after a form of *bać*, separated from that form by at most a single personal pronoun:[5]

(4.48)
```
[orth=się][orth!="[.!?,:]"]{,5}[base=bać]
|[base=bać][base="on|ja|ty|my|wy"]?[orth=się]
```

4.1.4. Searching for tags

The rather baroque query (4.48) can be simplified by replacing the condition `orth!="[.!?,:]"` with a direct reference to the 'grammatical class' interp (cf. ch. 3):

(4.48′)
```
[orth=się][pos!=interp]{,5}[base=bać]
|[base=bać][base="on|ja|ty|my|wy"]?[orth=się]
```

In general, the values of the `pos` attribute are the abbreviations of names of grammatical classes discussed in §3.4 (cf. the table on p. 36). For example, a query about a sequence of two nominal forms beginning with an *a* may be formulated as follows:

(4.49) `[pos=subst & orth="a.*"]{2}`

[5] This query is broken into two lines for typographic reasons.

The specifications of the values of `pos` may, just as in case of `orth` and `base`, contain regular expressions. For example, taking into account the fact that personal pronouns are split between the class of 3rd person pronouns ppron3 and non-3rd person pronouns ppron12, the following queries may be used to find any form of any personal pronoun:

(4.50) `[pos=ppron12 | pos=ppron3]`

(4.51) `[pos="ppron12|ppron3"]`

(4.52) `[pos="ppron(12|3)"]`

(4.53) `[pos="ppron[123]+"]`

(4.54) `[pos="ppron.+"]`

(4.55) `[pos=ppron/x]`

That means that the query (4.48) may be further simplified:

(4.48″)
```
[orth=się][pos!=interp]{,5}[base=bać]
|[base=bać][pos=ppron/x]?[orth=się]
```

Apart from the specifications of segments (with the help of `orth`), base forms (`base`) and grammatical classes (`pos`), queries may contain specifications of particular grammatical categories, such as case or gender. The following attributes may be used to this end (cf. §3.3):

attribute	**possible values**
number	sg pl
case	nom gen dat acc inst loc voc
gender	m1 m2 m3 f n
person	pri sec ter
degree	pos comp sup
aspect	imperf perf
negation	aff neg
accentability	akc nakc
post-prepositionality	npraep praep
accommodability	congr rec
agglutination	agl nagl
vocalicity	nwok wok

Hence, it is possible to pose the following queries:

(4.56) `[number=sg]`
find singular forms

(4.57) `[pos=subst & number=sg]`
find singular nominal forms

(4.58) `[pos=subst & gender!=f]`
find masculine and neuter nominal forms

(4.59) `[number=sg & case="nom|acc" & gender="m[123]"]`
find singular masculine forms in the nominative or in the accusative case

The following three-letter abbreviations may be used instead of the full names of the attributes:

attribute	**abbreviation**
number	nmb
case	cas
gender	gnd
person	per
degree	deg
aspect	asp
negation	neg
accommodability	acm
accentability	acn
post-prepositionality	ppr
agglutination	agg
vocalicity	vcl

For example, the query below is equivalent to (4.59):

(4.59′) `[nmb=sg & cas="nom|acc" & gnd="m[123]"]`

In the graphical and text versions of Poliqarp, it is possible to define so-called aliases, i.e., abbreviations for alternative values of a given attribute, which may themselves be used as if they were possible values of attributes. The current version of the IPI PAN Corpus has four such aliases already pre-defined:

alias	definition
masc	m1 m2 m3
noun	subst depr ger xxs ppron12 ppron3
pron	ppron12 ppron3 siebie
verb	fin praet aglt bedzie inf imps impt
	pact ppas pcon pant ger winien

With the definitions of the aliases `noun` and `masc` given above, the following two queries are equivalent:

(4.60) `[pos=noun & gender=masc]`

(4.61) `[pos="subst|depr|ger|xxs|ppron12|ppron3" & gender="m1|m2|m3"]`

The values of grammatical classes and categories may be specified jointly, with the use of the `tag` attribute. For example, the following query may be used to find singular nominative neuter nouns:

(4.62) `[tag=subst:sg:nom:n]`

The values of the `tag` attribute have the form $\texttt{kl:kat}_1\texttt{:kat}_2\texttt{:...:kat}_n$, where `kl` is the name of a grammatical class, while each of $\texttt{kat}_i$ is the value of a grammatical category appropriate for that class, in the order specified in the table on p. 31 (§3.4.2).

Just as in case of other attributes, also the specification of the value of `tag` may contain regular expressions, e.g.:

(4.59″) `[tag=".*:sg:(nom|acc):m[123].*"]`

(4.59‴) `[tag="sg:(nom|acc):m[123]"/x]`

4.1.5. Ambiguities

One of the features that distinguish the IPI PAN Corpus and Poliqarp from other corpora and search tools is the representation and processing of ambiguities. As discussed, e.g., in Oliva 2001, and as already mentioned in §2.2, there are cases where it is impossible to tell which of a number of interpretations is the right one, as in (2.5) in §2.2, repeated below.

(2.5) Pamiętam ją pijaną.
remember.1ST her.ACC drunk.ACC/INS
'I remember her drunk.'

(2.6) a. Pamiętam go pijanego.
remember.1ST him.ACC drunk.ACC
'I remember him drunk.'

b. Pamiętam go pijanym.
remember.1ST him.ACC drunk.INS

Since it is impossible to resolve the grammatical case of *pijaną* in (2.5), both interpretations, accusative and instrumental, should be marked in the corpus as correct in this context.

However, given that after disambiguation a single segment may contain more than one interpretation, the question arises whether such ambiguous segments, e.g., *pijaną* in (2.5), should be included in the result of a query which matches only some of these interpretations, e.g., in the result of the query `[case=acc]`. On the one hand, the segment *pijaną* should be included in the result of `[case=acc]`, as accusative is one of the correct interpretations of this segment in this context, but on the other hand, this segment should *not* be included, as it is not absolutely certain that this is an accusative form.

Instead of choosing between these interpretations of a query like `[case=acc]`, Poliqarp allows the user to pose both kinds of queries. When a single equality sign is used, as in `[case=acc]`, all segments whose at least one interpretation matches the given condition will be returned, so both *pijaną* and *ją* in (2.5) will be included in the result of this query. On the other hand, when two equality signs are used, as in `[case==acc]`, only those segments will be returned whose all interpretations satisfy the condition expressed with `==`, i.e., in (2.5), only the form *ją* will match the query.

With this distinction in hand, it is possible to search for forms which, e.g., may in a given context be interpreted as either accusative or genitive (cf. (2.3) on p. 15), so — given a properly tagged corpus — the following query should give non-empty results.

(4.63) `[case=acc & case=gen]`

Conversely, the query below matches those segments whose all interpretations in the given context are at the same time accusative and genitive, so it will necessarily produce empty results.

(4.64) `[case==acc & case==gen]`

The queries above pertain to interpretations which are the result of morphosyntactic disambiguation. As mentioned in §2.2 above, the IPI PAN Corpus contains also all other interpretations assigned to a given segment by the morphological analyser.

In some situations it is useful to have access to such interpretations rejected by the disambiguator, e.g., for the task of finding all syncretic forms of a certain kind in the corpus, or when investigating disambiguation errors. For example, in order to find all syncretic accusative/genitive forms in the corpus, regardless of their interpretation in contexts in which they occur, the following query may be posed:

(4.65) `[case~acc & case~gen]`

The final equality operator available in Poliqarp queries is ~~. The following query may be used for finding those forms which are unambiguously accusative, again, regardless of the context in which they occur.

(4.66) `[case~~acc]`

The table below summarises the four equality operators put at the user's disposal in Poliqarp.

	in the results of morphological analysis	**in the results of disambiguation**
at least one interpretation	~	=
each interpretation	~~	==

It should be clear that the following implications hold:

- `[attribute~~value]` → `[attribute==value]`
  i.e., each match of the query `[attribute~~value]` also matches the query `[attribute==value]`
- `[attribute==value]` → `[attribute=value]`
  i.e., each match of the query `[attribute==value]` also matches the query `[attribute=value]`
- `[attribute=value]` → `[attribute~value]`
  i.e., each match of the query `[attribute=value]` also matches the query `[attribute~value]`

4.1.6. Constraining matches to sentences or paragraphs

Texts contained in the IPI PAN Corpus are divided into sentences and paragraphs. This information may be taken into account in queries, in order to constrain a query to a sentence or a paragraph, as in the query below, which may be used to find the form *się* separated from a form of the verb BAĆ by any positive number of (non-*się*) segments, but within a sentence.

(4.67) `[base=bać][orth!=się]+[orth=się] within s`

Similarly, the qualifier '`within p`' constrains the scope of a query to a paragraph.

4.1.7. Constraining matches with metadata

Each text in the IPI PAN Corpus comes with a set of data about that text, such as its title and author, publisher, date of publication, etc. Some of such metadata are accessible through Poliqarp and may be used to constrain the scope of a query, e.g., to texts by a given author or published between certain dates.

There are three types of metadata available in the current version of the IPI PAN Corpus, and there are five meta-attributes which correspond to those three types:

- the name of the author or authors: the attribute `author`,
- the title: the attribute `title`,
- date of creation or publication: three attributes:
  - `created` — when the text was created,
  - `first_published` — when it was first published,
  - `published` — when the version in the corpus was published.

Usually only some of these attributes will have a value defined, e.g., when only the date of the publication is known, not the date of the first publication or the date of origin, or in case of short newspaper notes, which might lack information about the author or even the title.

In order to constrain the scope of a query with metadata, the keyword `meta` should be placed at the end of the query and it should be followed by

specifications of values of meta-attributes. In case the scope of the query is also constrained to a sentence or to a paragraph, the specification of metadata should follow the structural constraint, e.g.:

(4.68) `[pos=subst]{6,} within s meta author=Kowalski`

Just as in case of ordinary attributes such as `orth` or `pos`, also the specifications of values of the meta-attributes `author` and `title` may contain regular expressions. For example, the query below may be used to find forms of the lexeme WIRUS in those texts whose title contains one of the sequences: *windows* or *microsoft*.

(4.69) `[base=wirus] meta title="windows|microsoft"`

By default, the specifications of values of `author` and `title` are taken to be case-insensitive and they are interpreted as matching (at least) parts of values of appropriate meta-attributes, so the following query will find sequences of nominal forms in works by, *inter alia*, Pol, Polkowski and Rampolski:

(4.70) `[pos=subst]{5} meta author=Pol`

To change that default behaviour, the flags `/X` and `/I` may be used. The effect of these flags is dual to the effect of the flags `/x` and `/i` described above: the effect of `/X` is that a given specification of the value of an attribute is understood as matching the complete value of that attribute, while the flag `/I` enforces the case-sensitive interpretation, as in examples below:

(4.71) `[pos=subst]{5} meta author=Pol/I`
the scope of the query is limited to texts by Pol, Polkowski, etc., but not by Rampolski,

(4.72) `[pos=subst]{5} meta author="Marek Pol"/X`
query limited to texts by Marek Pol,

(4.73) `[pos=subst]{5} meta author=Pol/X`
query limited to texts by those authors whose complete name is either *Pol*, or *POL*, or *pol*, etc.

(4.74) `[pos=subst]{5} meta author=".* Pol"/I`
query limited to texts by an author whose surname is *Pol*.

Regular expressions are not allowed in case of the date-valued attributes `created`, `first_published` and `published`. On the other hand, it is possible to use the lesser/greater signs `<` and `>`, e.g.:

(4.75) `[pos=subst]{5} meta created>1950`
query limited to texts created after year 1950

Constraints on meta-attributes may be combined with the operators `&`, `|` and `!`, e.g:

(4.76) `[pos=subst]{5} meta created>=1951 & created<=1960`
query limited to texts created in years 1951–1960

(4.77) `[pos=subst]{5} meta published>1900 & author!=Sienkiewicz`
query limited to texts published after 1900, by authors other than Sienkiewicz

(4.78) `[pos=subst]{5} meta (author=sienkiewicz & title=potop) | (author=żeromski & title=przedwiośnie)`
query limited to *Potop* by Sienkiewicz and *Przedwiośnie* by Żeromski

In the current preliminary version of the IPI PAN Corpus, many texts do not have complete metadata associated with them. The results of the queries involving metadata specifications above will only come from those texts which have values of the relevant meta-attributes defined. That means that, perhaps contrary to expectations, the result of the first of the two queries below will be a small subset of the result of the second query.

(4.79) `[pos=subst] meta created>1900 | created<=1900`

(4.80) `[pos=subst]`

In case a given meta-attribute does not have a value defined, it is assumed that its value is the empty string, i.e., `""`, so the query below is equivalent to query (4.80).

(4.81) `[pos=subst] meta published>1900 | published<=1900 | published=""`

4.1.8. Aligning matches

In order to make the results of a query more readable, it is possible to place within the query proper, i.e., before the qualifiers `within` and `meta`, a special alignment marker, ^, as in:

(4.82) `[pos=adj & case=nom]+^[pos=subst & case=nom]+`

Instead of the usual three columns containing the left context of the match, the match itself, and the right context, the results of this query will be split into four columns, containing, respectively, the left context, the left match, i.e., the sequence of segments matching the part of the query before the alignment marker ^ (here, a non-empty sequence of nominative adjectives), the right match (here, a non-empty sequence of nominative nouns), and the right match, as in Fig. 4.1.

4.2. Poliqarp

The aim of this section is to present three Poliqarp interfaces: from the WWW version (§4.2.1), with the most limited functionality, through the graphical version (§4.2.2), appropriate for most users, to the GNU/Linux text version (§4.2.3), quicker and more powerful than the previous two versions, but with the visually least attractive user interface.

4.2.1. The WWW version

The Internet version of the IPI PAN Corpus is available at `korpus.pl`. There are currently no restrictions on the access to the corpus, but such restrictions might be introduced in case the popularity of that service exceeds the capacity of the WWW server on which the corpus resides.

Although the functionality of the current WWW version is substantially limited in comparison with the other two interfaces, the full query syntax, as described in §4.1, is accepted by that version. After pointing the browser at `korpus.pl` and selecting the corpus, the user is presented with a query box, as in Fig. 4.2.

After entering the query and pressing Enter (or clicking on Execute), the results will be displayed in three or four columns, depending on whether the query contained the alignment marker, cf. Fig. 4.3. A larger context of

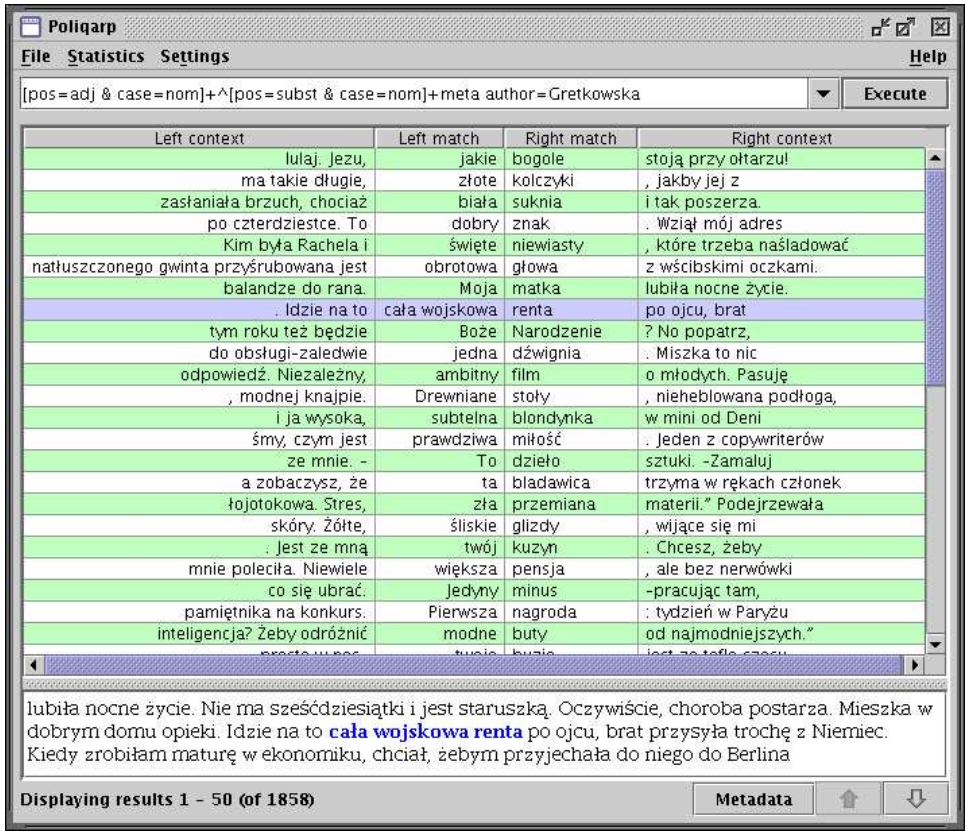


Figure 4.1. Match alignment in the graphical version

any given result may be displayed by clicking on the result, or — more precisely — on the central part of the result which shows the match, cf. Fig. 4.4. The results of a query are shown in batches containing 25 results each. In order to view the next 25 results, the user should click on the button Next 25, while in order to go back to the previous 25 results, the user should click on Previous 25.

The only additional functionality present in the current WWW version is the option to sort the results according to the elements of any of the three or four columns, alphabetically *a fronte* (from the beginning of strings) or *a tergo* (from the end of strings), in the ascending or in the descending order.

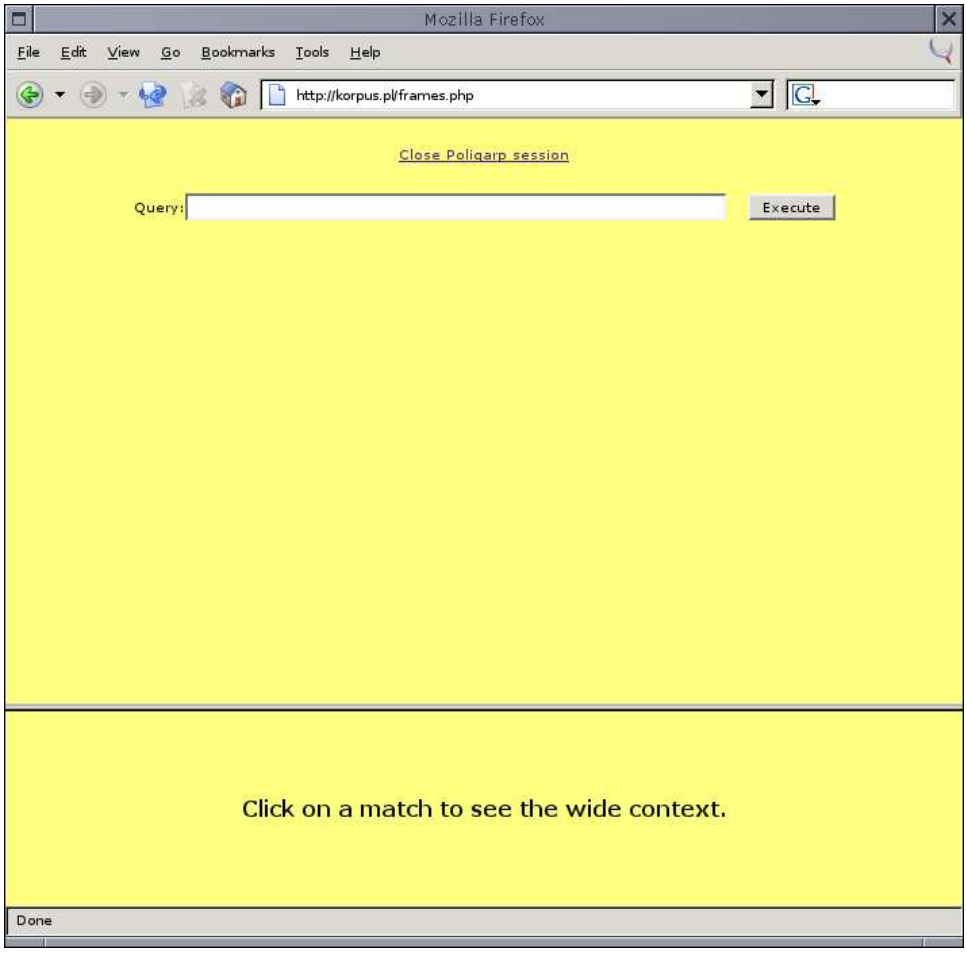


Figure 4.2. WWW version after selecting the corpus

The WWW version of Poliqarp is the version which currently undergoes most dynamic changes, so the version available at the time this publication sees the light of day may substantially differ from the version described here.

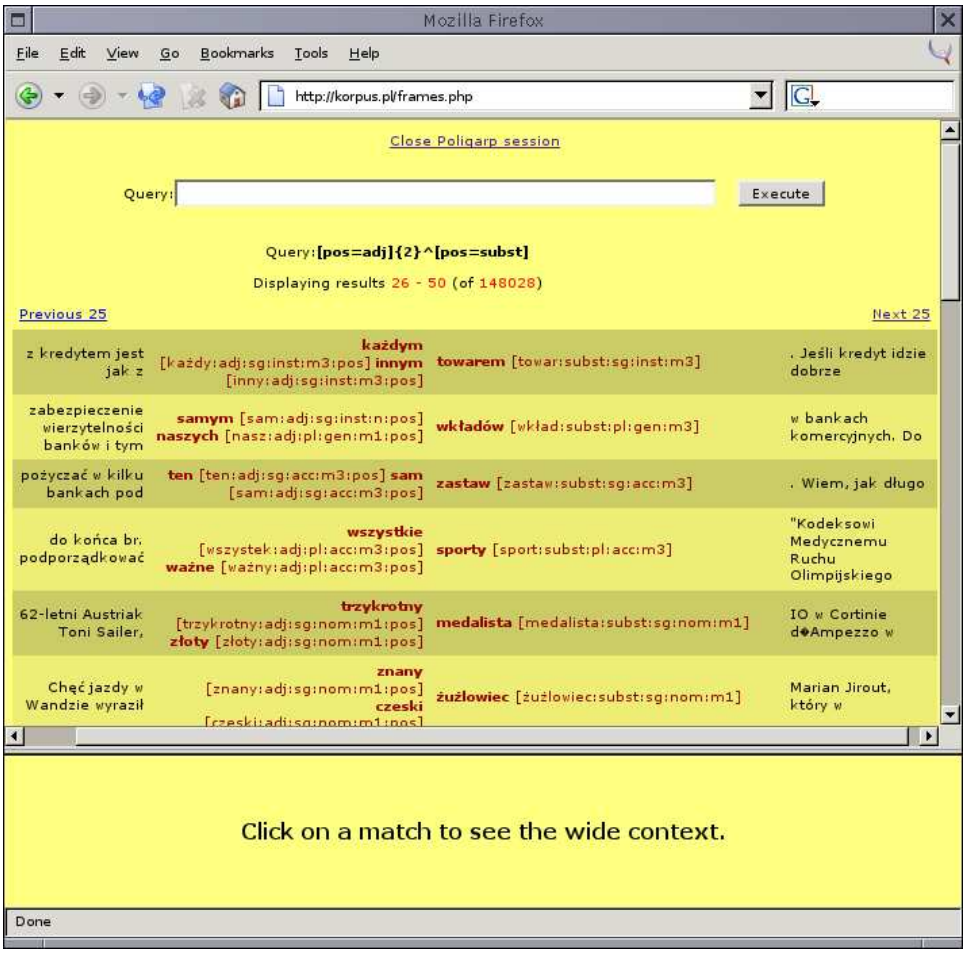


Figure 4.3. WWW version: results of a query

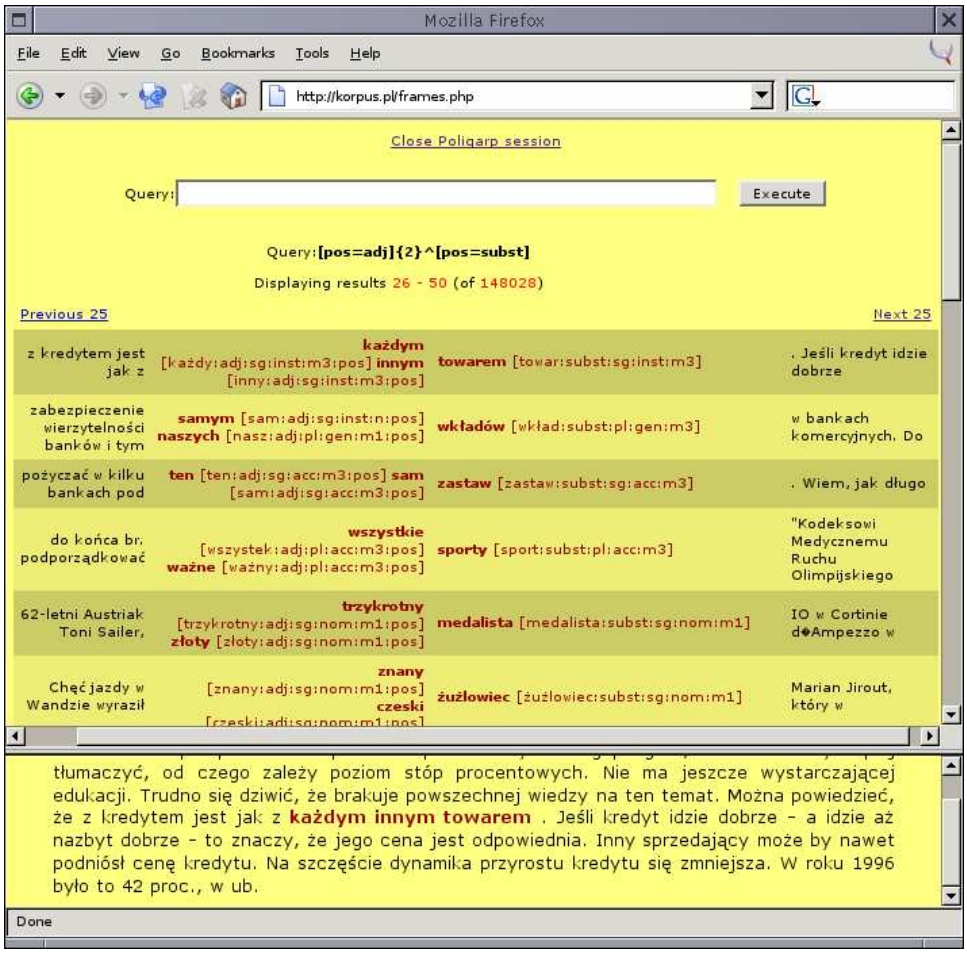


Figure 4.4. WWW version: larger context

4.2.2. The GUI version

The advertised version of the search engine and concordancer Poliqarp is the graphical version. Apart from the menu at the top of the window, the main window of the graphical version consists of the query box, the upper window for displaying the results of a query, and the lower window for displaying the larger context, as in Fig. 4.1 on p. 61, and for presenting the metadata.

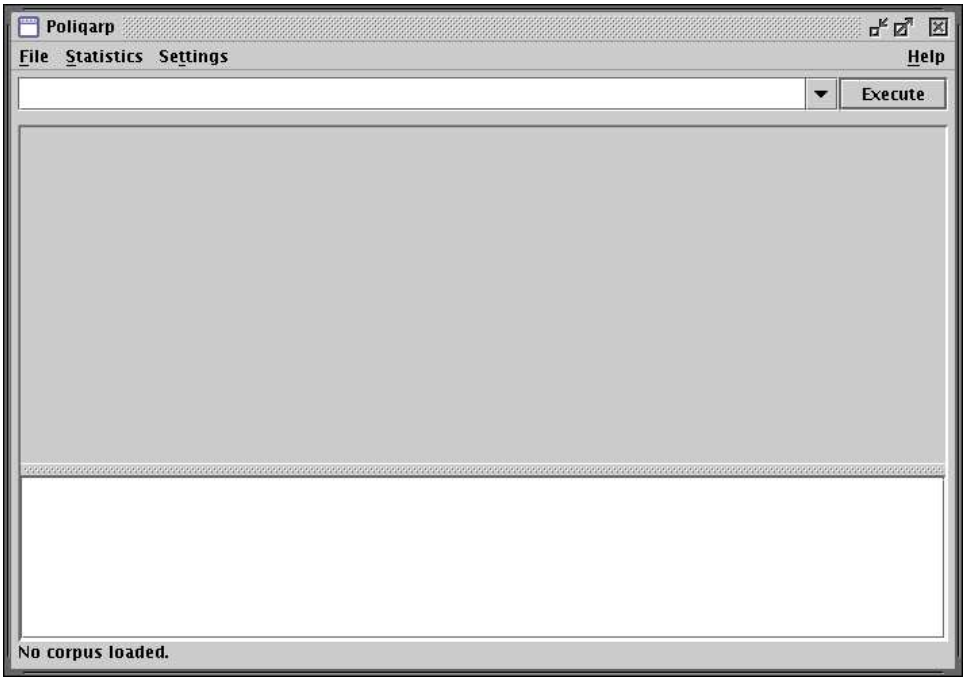


Figure 4.5. Graphical version immediately after start up

Immediately after starting Poliqarp, the query box and both window parts are empty, as shown in Fig 4.5. Before searching a corpus, it is necessary to load the corpus — this can be done by choosing File in the menu, then selecting Open corpus..., and then identifying the directory containing the IPI PAN Corpus on the CD-ROM or the hard disk, depending on where the corpus is installed (cf. appendix A). The same sequence of steps can be used to change the current corpus, without leaving the program.

Once Poliqarp loads the corpus, it stores its localisation, and the next time Poliqarp is started this localisation is available from the menu File, and then Most recently used. It is also possible to load the first of the corpora whose localisations have been stored by pressing Ctrl-Alt-1, the second — by pressing Ctrl-Alt-2, etc.

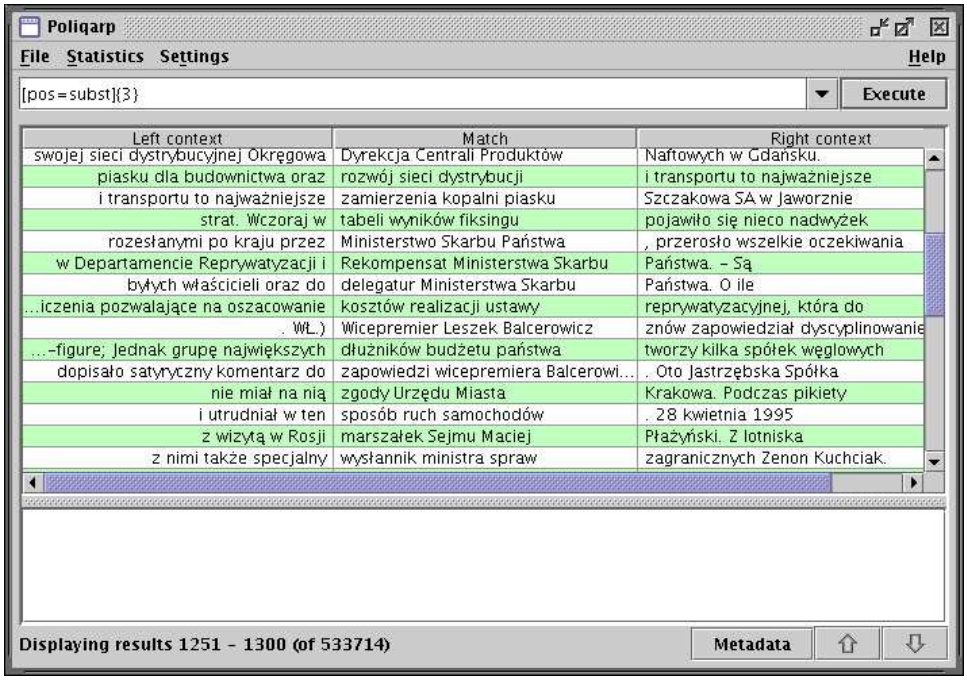


Figure 4.6. Query results without alignment

After loading the corpus, the user may enter the query in the query box, and then press Enter or click on Execute to actually start the search. If the query did not contain the alignment marker, three columns will be shown in the upper window, containing, for each query result, the left context, the match and the right context, as shown in Fig. 4.6. Otherwise, four columns will be displayed, as explained in §4.1.8. The width of the columns may be changed by dragging the borders between the headers of the columns with the mouse.

Poliqarp stores the history of queries posed in the current session, as well as in the previous sessions. The history is accessible, e.g., by clicking on the little button placed between the query box and the Execute button.

After executing a query, the first fifty results will be shown in the upper window. In order to view subsequent results, the down arrow in the bottom right corner of the Poliqarp window should be clicked on. Clicking on the up arrow placed to the left from the down arrow will re-display previous results.

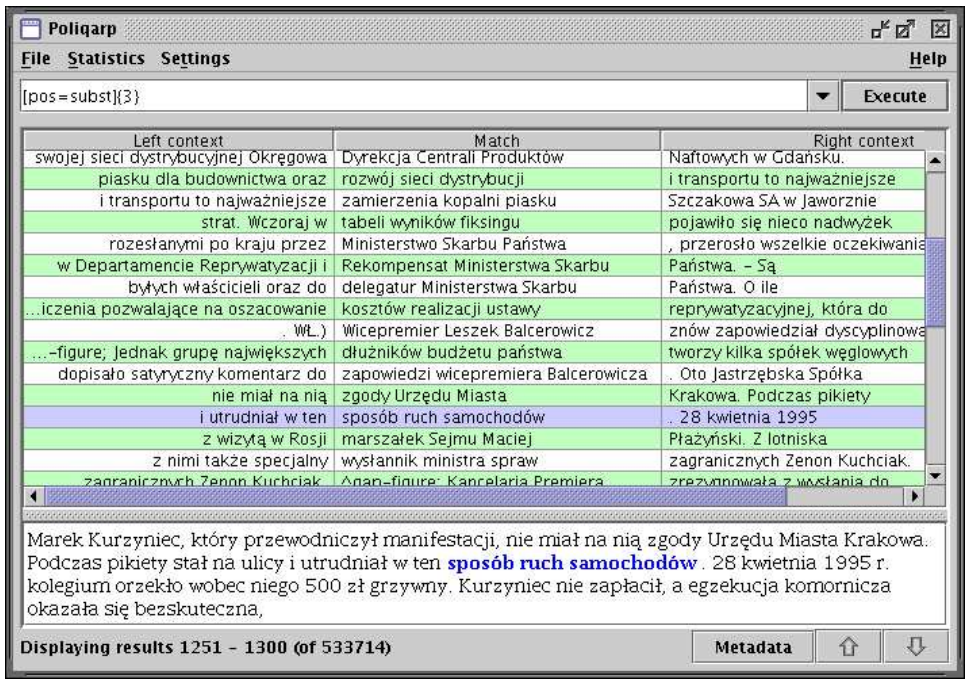


Figure 4.7. Larger context in the graphical version

After clicking on any of the results in the upper window, a larger context of that result will be displayed in the lower window. The lower window is also used to present metadata — in order to switch between the context mode and the metadata mode of the lower window, the button marked as Metadata (in the larger context mode) or Context (in the metadata mode) should be pressed.

Query results can be sorted by any of the columns. In the simplest case, in order to sort the results alphabetically *a fronte* in the ascending order, it is sufficient to click with the left mouse button on the header of the column, e.g., on the header Match (in case the query did not contain the alignment marker). Clicking on that header again will have the effect of re-sorting the results in the descending order. Clicking again will restore the ascending order, etc.

It is also possible to sort columns *a tergo*, i.e., from the end, both in the ascending and in the descending order. This functionality is available from a menu which is invoked by clicking on the appropriate header with the right hand button of the mouse, as shown in Fig. 4.8, which illustrates the *a tergo* ascending sorting of the left match column.

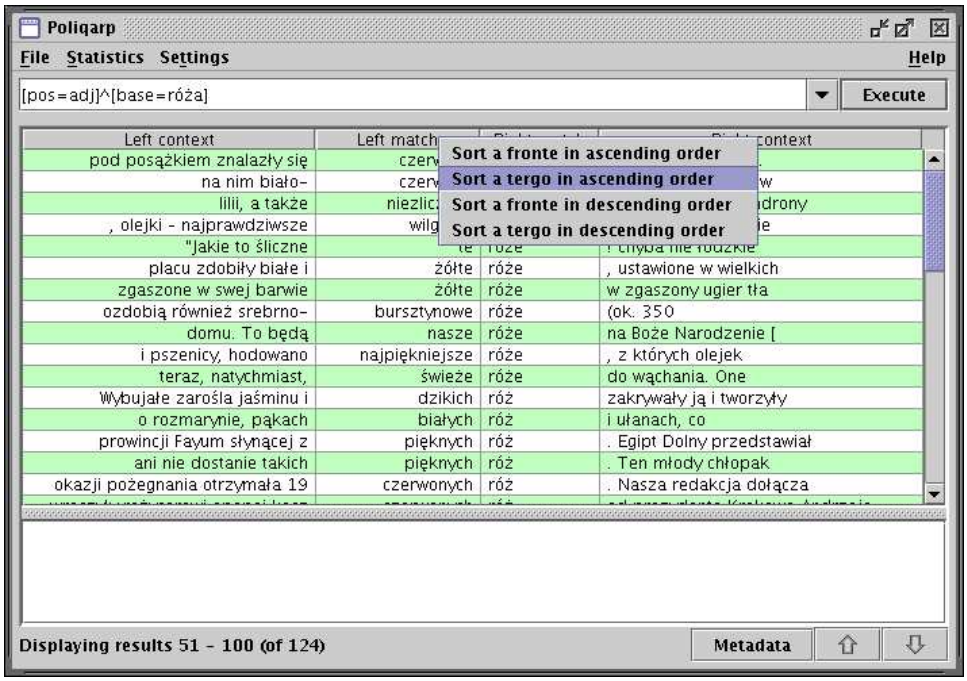


Figure 4.8. Sorting *a tergo* in the ascending order

The way results should be sorted can also be set globally, from the menu Settings, by selecting the submenu Options... and then the Sorting

tab. Settings of the sorting options changed this way are binding for the following queries and they are stored for future sessions.

One of the customisation options available from Settings → Options... is Context, which allows the user to specify the size (in segments) of the left context column, the right context column, and the larger context presented in the lower window.

An important feature of Poliqarp and other similar search and concordance engines is the possibility to view not just the orthographic forms (the segments), but also their base forms and morphosyntactic disambiguated tags. The current graphical version of Poliqarp allows the user to determine, separately for the context columns and for the match column(s), which of these three kinds of information (segments, base forms, tags) should be displayed. For example, the settings shown in Fig. 4.9 correspond to the formatting of results illustrated in Fig. 4.10.

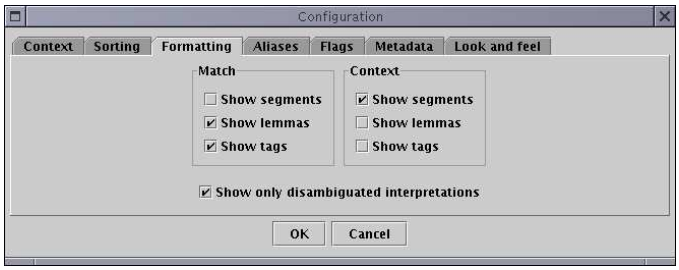


Figure 4.9. Customising the display options

As mentioned above (p. 53), the graphical version of Poliqarp lets the user define aliases, i.e., abbreviations of alternative values of a given attribute. Aliases can be added, edited and removed via the Aliases tab in Settings → Options... In the current version of the program, aliases defined this way are not persistent, i.e., they are not stored for future sessions.

Another tab, Flags, allows the user to determine, separately for the query proper and for the meta part of the query, whether specifications of attribute values should be understood as pertaining to the complete value, or to its part, and whether they should be understood in the case-sensitive manner.

By default, a query like `[pos=ppro]` will match those segments whose grammatical class is exactly ppro, i.e., it will return the empty result. In

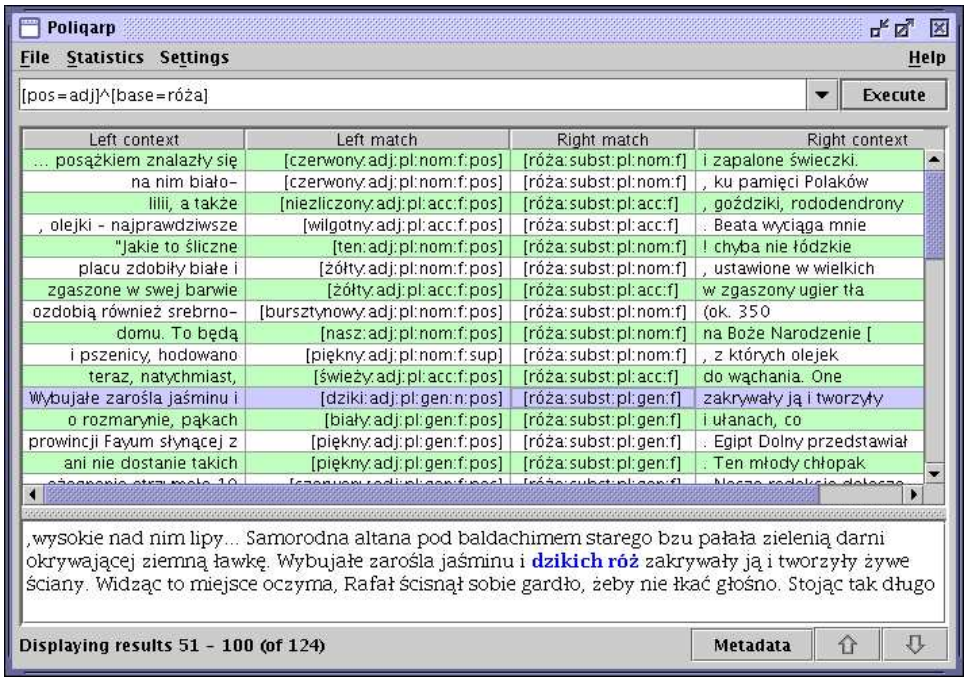


Figure 4.10. Query results displayed according to the options in Fig. 4.9

order to find forms of personal pronouns, whose actual grammatical class is either ppron12 or ppron3, that query should also include the flag `/x`, as discussed above (p. 47). In order to make Poliqarp interpret all conditions as if they contained the flag `/x`, the option Whole words only in the Query column of the Flags tab should be checked.

Similarly, it is possible to change the complete match vs. partial match interpretation of query conditions on metadata, and the case sensitivity of attribute value specifications in the query proper, and in the conditions on metadata.

Another tab in the Settings → Options... menu is the Metadata tab. It allows to the user to constrain the scope of a series of queries to texts with certain values of meta-attributes, e.g., to texts by a given author or published within a given period. Finally, the Look and feel tabs lets the user select the size of the font.

In the main Poliqarp menu, apart from the submenus File and Settings, there is also another submenu, Statistics. In the current version of Poliqarp, this submenu contains only the most rudimentary quantitative information about the corpus. For example, the information displayed in Fig. 4.11, concerning a subcorpus of the IPI PAN Corpus, says that there are over 56 million positions (tokens) in that subcorpus, occupied by over 729 thousand different segments (types) with over 355 thousand different lemmata. The number of different tags (sequences of the form `kl:kat`$_1$`:kat`$_2$`:...:kat`$_n$, cf. p. 54) in this corpus is 1150.

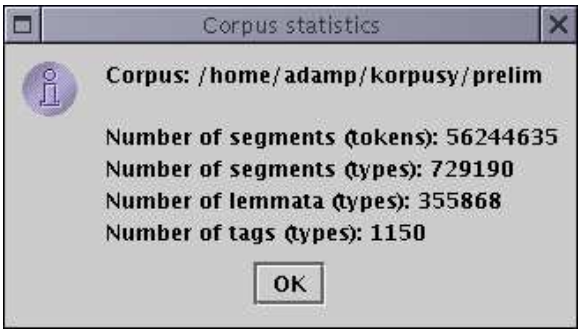


Figure 4.11. Quantitative information about a corpus

Finally, Save results... in the File menu saves the results of the last query, while Exit starts the process of erasing the contents of the hard disk.[6]

4.2.3. The text version

The text version of Poliqarp is designed for PC computers running the GNU/Linux operating system.

The program can be started with the command `poliqarp corpus`, where `corpus` is the name of the corpus (the first segment in the names of files such as `wstepny.cfg` and `wstepny.poliqarp.corpus.image`), including the path to that corpus. For example, assuming the corpus is located in the `./corpus/` directory and consists of files such as `wstepny.cfg`, `wstepny.poliqarp.chunk.image`, etc., the program can be invoked in the following way:

[6] Just kidding.

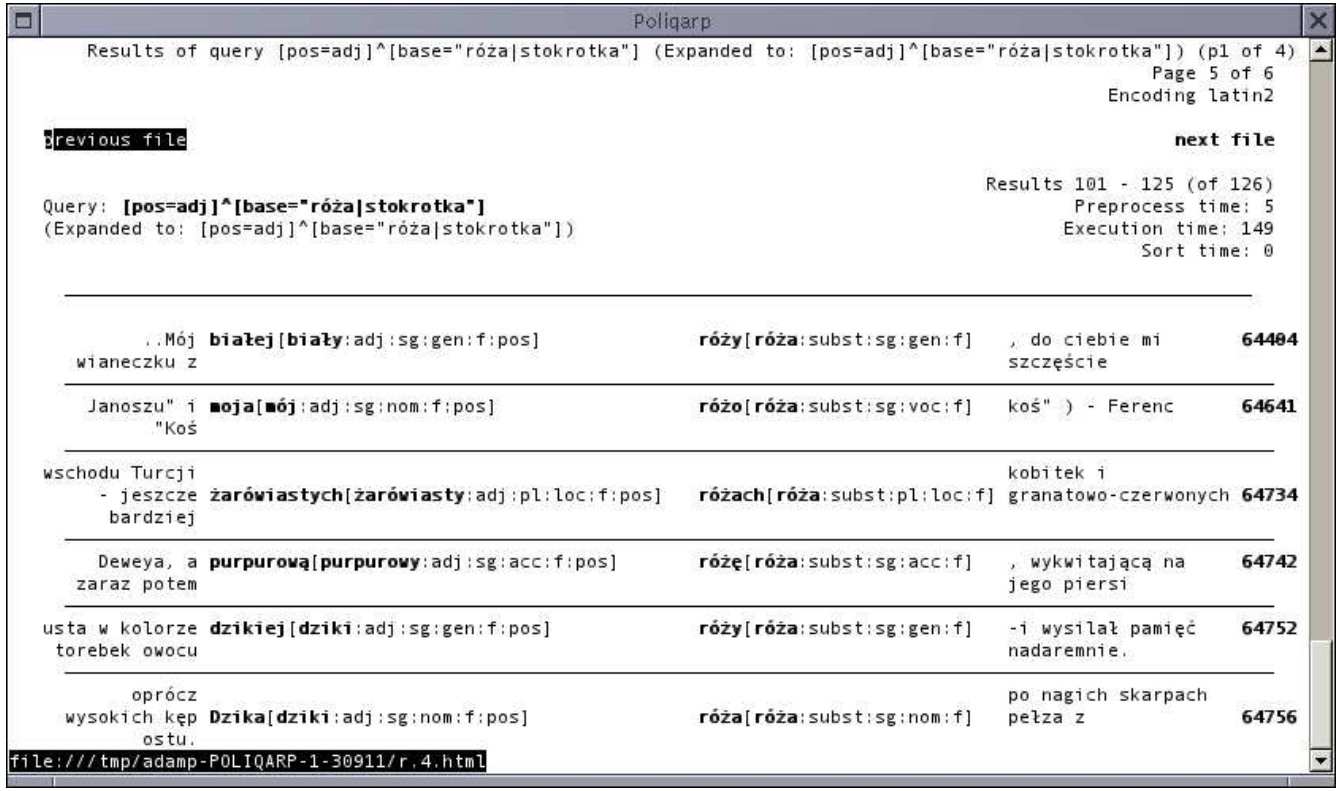


Figure 4.12. Query results in the text version (default configuration)

(4.83) `$ poliqarp corpus/wstepny`

As a result of executing that command, the shell command line will be replaced by the Poliqarp command line, e.g.:

(4.84) `CORPUS/WSTEPNY>`

The Poliqarp command line can be used both to enter queries and to enter Poliqarp commands. In order to pose a query, the user should enter the query in the command line and press Enter. Results of a query like (4.85) will be displayed in the format shown in Fig. 4.12.

(4.85) `CORPUS/WSTEPNY> [pos=adj]^[base="róża|stokrotka"]`

Unless the user specifies otherwise, query results will be generated in the HTML format and they will be displayed by the Links browser (which, hence, should be installed in the system).

4.2.3.1. Editing queries on the command line

Just as in case of the graphical version of the program, also the text version of Poliqarp stores query history, which also includes commands issued from the Poliqarp command line. These past queries and commands may be accessed by pressing the arrow up and arrow down cursor keys.

There is also a limited history search functionality: pressing the keys Ctrl-r and entering a sequence of characters (and then Enter) results in invoking the last query or command in the history which contains that sequence of characters.

Queries can be edited using the usual cursor keys, the Backspace key, as well as, *inter alia*, key combinations known from Emacs, as illustrated in the table on the next page. On the other hand, the keys Del, Home and End do not have the expected effect.

When sending the query for processing (by pressing Enter), the cursor may be placed in any position on the command line, not necessarily at the end of the line.

key	effect
Ctrl-b	moves the cursor one character to the left
Ctrl-f	moves the cursor one character to the right
Ctrl-a	moves the cursor to the beginning of the line
Ctrl-e	moves the cursor to the end of the line
Esc-b	moves the cursor one word to the left
Esc-f	moves the cursor one word to the right
Ctrl-d	deletes the character at the cursor position
Esc-d	deletes characters from the cursor to the end of the current word
Esc-Backspace	deletes characters from the beginning of the word to the character immediately preceding the cursor
Ctrl-k	deletes characters from the cursor to the end of the line
Esc-4 Ctrl-f	same effect as pressing Ctrl-f four times, i.e., moves the cursor four characters to the right
Esc-2 Esc-d	same as pressing Esc-d twice
	etc.

4.2.3.2. Customisation

The operation of the text version of Poliqarp may be changed by modifying the configuration file `.poliqarp_config`, placed in the user's home directory. The simplest way of creating the first version of such a configuration file is to issue, at the Poliqarp command line, the command `/dump-config`. This will result in the creation of a file called `.poliqarp_config.new`, also placed in the user's home directory, containing the default settings of the text version of Poliqarp. In order for this file to become the valid configuration file, its name should be changed to `.poliqarp_config`.

The configuration file is, actually, just a sequence of Poliqarp commands, which are executed at startup. Each of these commands, if preceded with the slash character `/`, may also be issued directly from the Poliqarp command line.

For example, in order to change the size (in segments) of the right context for the purposes of the current session only, the following command should be entered at the Poliqarp command line:

(4.86) `CORPUS/WSTEPNY> /set right-context 20`

The configuration file analog of that command, but taking effect in subsequent Poliqarp sessions, would be:

(4.87) `set right-context 20`

Similarly, the size of the left context may be modified either from the command line, as in (4.88) (the change will only be valid during the current session), or putting the line (4.89) in the configuration file (the change will be valid for the subsequent sessions).

(4.88) `CORPUS/WSTEPNY> /set left-context 20`

(4.89) `set left-context 20`

Poliqarp variables `right-context` and `left-context` are just two of a number of variables modifiable with the `set` command. Brief descriptions of all such variables will be shown upon the execution of the `/desc` command, issued from the command line,[7] while the current values of all Poliqarp variables may be examined with the `/show` command. If any of these two commands is invoked with arguments, which are the names of some of Poliqarp variables, the appropriate information will be displayed only for those variables. For example, in order to obtain the information about the current values of `right-context` and `left-context`, the following command should be given:

(4.90) `CORPUS/WSTEPNY> /show right-context left-context`

As Fig. 4.12 illustrates, left and right context columns contain, by default, only the segments, i.e., orthographic forms as they occur in the text, while the match columns contain also the base forms and the disambiguated tags. This behaviour may be modified by setting the values of the following variables: `cl-x` (to change the kind of information displayed in the left context), `ml-x` (left match), `mr-x` (right match or, in case there is only one match column, the whole match) and `cr-x` (right context). The values of these variables are sequences of letters. If one of these letters is `o` (as in `orth`), the information about the segments (orthographic forms)

[7] Just as all the other commands of the text version, also the `desc` command may be placed in the configuration file — this will result in the brief description being displayed at each subsequent startup of the text version.

will be among the information shown in this column. If `b` (as in `base`) is one of these letters, the base form will be displayed. Finally, if the value of the variable contains the letter `t` (as in `tag`), morphosyntactic tags after disambiguation will be displayed, while in case it contains the capital letter `T`, all tags assigned by the morphological analyser will be shown. For example, assuming the configuration file contains the lines shown in (4.91), the result of a query will be formatted as in Fig. 4.13.

(4.91)
```
set cl-x bo
set ml-x bt
set mr-x bt
set cr-x o
```

In the current text version of Poliqarp, query results are generated as a sequence of HTML files, each containing at most `hits-per-page` results. These results are displayed via a browser specified in the value of `pager`. For example, after the `set` commands below are executed, subsequent query results in this session will be displayed in the Mozilla browser, 50 results per page.

(4.92)
```
CORPUS/WSTEPNY> /set hits-per-page 50
```

(4.93)
```
CORPUS/WSTEPNY> /set pager mozilla
```

The exact format of HTML results may be modified in detail by setting the variables `result-format` i `match-format`: the former determines the overall HTML format of the page, while the latter determines the format of a single result. The values of these variables should only be modified by users with a deep understanding of the HTML format. A detailed discussion of possible values of these two variables lies outside the scope of this publication.

Other variables that should be mentioned here include `input-encoding` and `output-encoding`: their values determine the character encoding assumed for the purpose of terminal input and output, respectively. Normally the values of these variables should not be modified. Their default value is `latin2`.

Similarly as in the graphical user interface, also the text version allows the user to sort the results according to the values of any of the columns, in the ascending or in the descending order, in the usual *a fronte* order, or in the *a tergo* order. The sorting order is specified via the `sort-by`

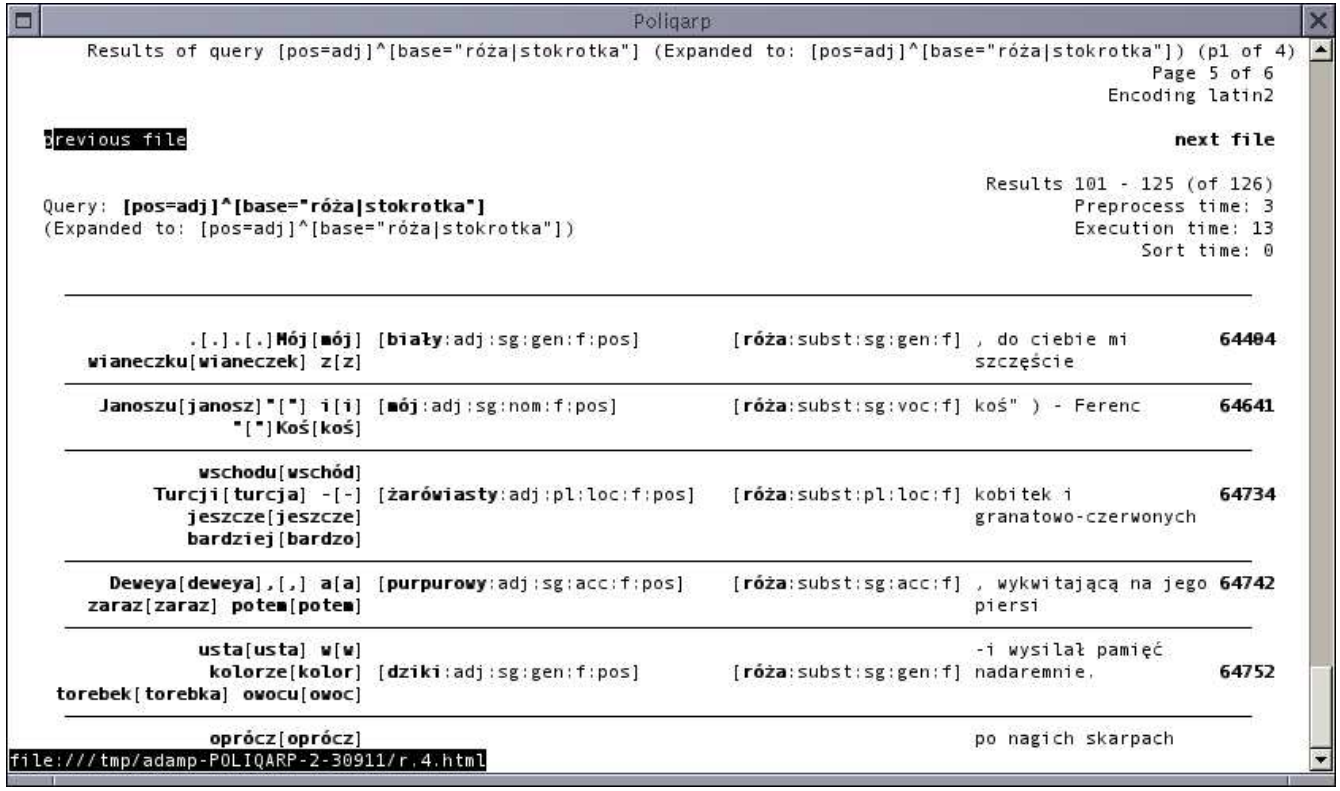


Figure 4.13. Formatting query results with the settings in (4.91)

variable, whose values are sequences of sort specifications. For example, after issuing the command in (4.94), query results will be sorted (*a fronte*, in the ascending order) according to the values of the right context.

(4.94) `CORPUS/WSTEPNY> /set sort-by r`

There is a different letter corresponding to each column:

letter	column
`l`	left context
`r`	right context
`n`	left match
`m`	right match

In case the value of the `sort-by` variable contains the small letter `l`, `r`, `n` or `m`, the results will be sorted in the usual *a fronte* order of the values of that column; if it contains the capital letter `L`, `R`, `N` or `M`, the results will be sorted in the *a tergo* order. Moreover, each of the letters may be preceded by `+` (for the ascending order) or `-` (for the descending order). The lack of `+` or `-` before a letter is interpreted as if `+` were present. For example, the effect of the command in (4.95) will be that subsequent results will be sorted according to the left context, *a tergo*, in the descending order.

(4.95) `CORPUS/WSTEPNY> /set sort-by -L`

If the value of the `sort-by` variable contains a number of sorting specifications referring to different columns, the results are first sorted according to the first specified column, then — in case values of that first specified column are equal — according to the second column specified in the value of `sort-by`, etc. For example, the command below will cause query results to be sorted in the ascending order according to the values of the right match column, and then, within this order, in the ascending order according to the left match column, as illustrated in Fig. 4.14.

(4.96) `CORPUS/WSTEPNY> /set sort-by m-n`

The results of the last query are remembered until the next query is posed and they can be re-sorted and re-displayed in the new order, without the need to execute the query again. The command that sorts the results according to the current sorting specifications is `sort`, while the command `view` displays the results. For example, in order to view

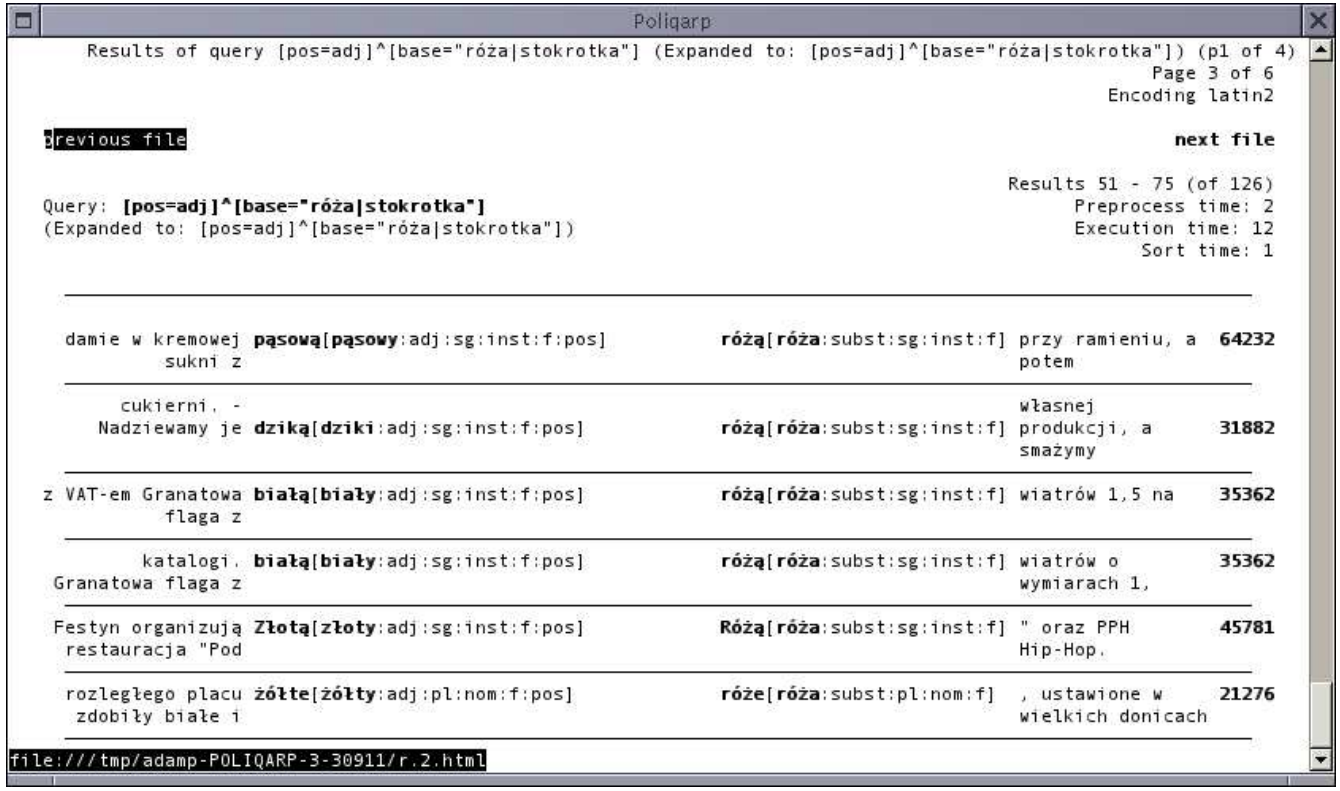


Figure 4.14. Sorting query results according to (4.96)

the results of the last query, re-sorted on the left context, the following sequence of commands should be given:

(4.97) `CORPUS/WSTEPNY> /set sort-by l`

(4.98) `CORPUS/WSTEPNY> /sort`

(4.99) `CORPUS/WSTEPNY> /view`

Instead of using the full variable names, it is possible to use their abbreviations, i.e., such prefixes of variable names which uniquely identify the variables. For example, instead of the command '`set cl-x bo`' in (4.91), a shorter command '`set cl bo`' may be given, where the name of the variable `cl-x` is abbreviated to `cl`; on the other hand, the command '`set c bo`' is not a valid command, as the abbreviation `c` does not uniquely identify the variable: it is not clear whether it is meant to refer to `cl-x`, or to `cr-x`.

Again similarly as in case of the graphical interface, also the text version allows the user to view metadata associated with particular search results. As shown in Fig. 4.12–4.14, the last column of the results table contains a number — this number identifies the text a given result is part of. In order to examine the metadata of that text, the `/meta <numer>` command should be issued at the Poliqarp command line, e.g., `/meta 10735`.

The text version also allows the user to define and cancel aliases. For example, once the following line is present in the configuration file, the sequence `ppron` can be used as an abbreviation for `ppron12` and `ppron3`.

(4.100) `alias ppron = ppron12 ppron3`

Of course, just as in case of other commands, this command can also be given at the Poliqarp command line, preceded by a slash. The `alias` command executed without any arguments will result in displaying all currently defined aliases, while the `unalias` command with one or more arguments, names of aliases, will result in the cancellation of these aliases.

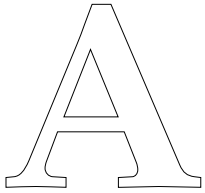
The command `exit` does what it says. It is also possible to change the current corpus without leaving Poliqarp: the `open` command should be used, as in the following example, which makes Poliqarp change the current corpus to `frek` in the `/home/adamp/corpus/` directory.

(4.101) `CORPUS/WSTEPNY> /open "/home/adamp/corpus/frek"`

Rudimentary quantitative information about the current corpus can be examined with the `stat` command, while the `help` command displays a compact description of all Poliqarp commands.

4.2.3.3. Environment variables

As in case of the graphical user interface, also the text version speaks two languages: Polish and English. The language is selected automatically, on the basis of the value of the system environment variable `LANG`: in case the value of that variable is `pl_PL`, the Polish interface to Poliqarp is started, otherwise, the English interface is used. Other standard environment variables which may influence the behaviour of Poliqarp are `HOME`, `TMP`, `TEMP` and `USER`. Their impact on Poliqarp is described in more detail in the `README.en.txt` file in the `linux` directory on the CD-ROM enclosed with this publication.



CD contents

The CD-ROM which is enclosed with the current publication contains:

- the preliminary version of the IPI PAN Corpus (the `corpus` directory),
- three versions of Poliqarp:
  - the graphical version designed for Windows 2000 and Windows XP (the `windows` directory and the startup script `autorun.inf`),[1]
  - the graphical version for GNU/Linux systems running on PCs (the `linux/gui` directory),
  - the text version for GNU/Linux systems running on PCs (the `linux/text` directory),
- this publication in the PDF format (the `pdf` directory).

This CD-ROM also contains the licence agreement (the `eula.en.txt` file) which defines the terms and conditions under which current versions of the IPI PAN Corpus and Poliqarp are distributed. Installing the IPI PAN Corpus and/or Poliqarp on a hard disk, starting Poliqarp, or redistributing the IPI PAN Corpus and/or Poliqarp in any form is tantamount to accepting the conditions of this licence agreement.

[1] This version should also work under the Windows 98 and Windows NT systems, but it was not tested under these systems.

A.1. Windows

After inserting the CD-ROM into a drive controlled by Microsoft Windows, the Poliqarp install wizard will be automatically started. The wizard will allow the user to select the directory in which Poliqarp should be installed, and it will offer the user the possibility of copying the corpus from the CD-ROM to the hard disk. Copying the corpus to hard disk is not necessary, but it will speed up corpus access.

The graphical version of Poliqarp, including the graphical version for Windows, requires an operational Java environment. The install wizard will attempt to find Java on the user's computer, and in case it fails, it will offer to install the Java version enclosed on the CD-ROM.

As a result of a successful installation of Poliqarp, it will be added to the Programs menu and, in case the user opted for that during the installation, an icon will be placed on the Desktop.

A.2. GNU/Linux

The installation of both the graphical and text interfaces of the GNU/ /Linux version of Poliqarp is described in the `README.en.txt` file in the `linux` directory.

Bibliography

Bański, P. (2001). The proposed annotation scheme for the IPI PAN corpus. IPI PAN Research Report 936, Institute of Computer Science, Polish Academy of Sciences.

Bański, P. (2003). Anotacja zewnętrzna: wpływ architektury korpusu IPI PAN na efektywność jego tworzenia oraz wykorzystania. *Polonica*, **XXII–XXIII**, 77–91.

Bień, J. S. (1991). *Koncepcja słownikowej informacji morfologicznej i jej komputerowej weryfikacji*, volume 383 of *Rozprawy Uniwersytetu Warszawskiego*. Wydawnictwa Uniwersytetu Warszawskiego, Warsaw.

Bień, J. S. (2004). An approach to computational morphology. In M. A. Kłopotek, S. T. Wierzchoń, and K. Trojanowski, editors, *Intelligent Information Processing and Web Mining*, Advances in Soft Computing, pages 191–199. Springer-Verlag, Berlin.

Bień, J. S. and Saloni, Z. (1982). Pojęcie wyrazu morfologicznego i jego zastosowanie do opisu fleksji polskiej (wersja wstępna). *Prace Filologiczne*, **XXXI**, 31–45.

Christ, O. (1994). A modular and flexible architecture for an integrated corpus query system. In *COMPLEX'94*, Budapest.

Dębowski, Ł. (2001). Tagowanie i dezambiguacja morfologiczna. IPI PAN Research Report 934, Institute of Computer Science, Polish Academy of Sciences.

Dębowski, Ł. (2003). A reconfigurable stochastic tagger for languages with complex tag structure. In *Proceedings of* Morphological Processing of Slavic Languages, *EACL 2003*.

Dębowski, Ł. (2004). Trigram morphosyntactic tagger for Polish. In *Proceedings of IIS:IIPWM 2004*.

Erjavec, T., editor (2001). *Specifications and Notation for MULTEXT-East Lexicon Encoding*. Ljubljana.

Gruszczyński, W. and Saloni, Z. (1978). Składnia grup liczebnikowych we

współczesnym języku polskim. *Studia Gramatyczne*, **II**, 17–42.

Ide, N., Priest-Dorman, G., and Véronis, J. (1996). Corpus encoding standard. Ms., http://www.cs.vassar.edu/CES/.

Ide, N., Bonhomme, P., and Romary, L. (2000). XCES: An XML-based standard for linguistic corpora. In *Proceedings of the Linguistic Resources and Evaluation Conference*, Athens, Greece.

Kupść, A. (1999). Haplology of the Polish reflexive marker. In R. D. Borsley and A. Przepiórkowski, editors, *Slavic in Head-Driven Phrase Structure Grammar*, pages 91–124. CSLI Publications, Stanford, CA.

Kurcz, I., Lewicki, A., Sambor, J., and Woronczak, J. (1974). Słownictwo współczesnego języka polskiego. Listy frekwencyjne. Ms., University of Warsaw.

Kurcz, I., Lewicki, A., Sambor, J., Szafran, K., and Woronczak, J. (1990). *Słownik frekwencyjny polszczyzny współczesnej*. Wydawnictwo Instytutu Języka Polskiego PAN, Kraków.

Mańczak, W. (1956). Ile jest rodzajów w polskim? *Język Polski*, **XXXVI**(2), 116–121.

Oliva, K. (2001). On retaining ambiguity in disambiguated corpora. *TAL (Traitement Automatique des Langues)*, **42**(2).

Przepiórkowski, A. (2003a). A hierarchy of Polish genders. In P. Bański and A. Przepiórkowski, editors, *Generative Linguistics in Poland: Morpho-syntactic Investigations*, pages 109–122, Warsaw. Institute of Computer Science, Polish Academy of Sciences.

Przepiórkowski, A. (2003b). Składniowe uwarunkowania znakowania morfosyntaktycznego w korpusie IPI PAN. *Polonica*, **XXII–XXIII**, 57–76.

Przepiórkowski, A. (2004). Instrukcja konwertowania tekstów na format XML w projekcie korpusowym IPI PAN. Ms., Institute of Computer Science, Polish Academy of Sciences.

Przepiórkowski, A. and Woliński, M. (2003a). A flexemic tagset for Polish. In *Proceedings of* Morphological Processing of Slavic Languages, *EACL 2003*, pages 33–40, Lisbon.

Przepiórkowski, A. and Woliński, M. (2003b). The unbearable lightness of tagging: A case study in morphosyntactic tagging of Polish. In *Proceedings of the* 4th International Workshop on Linguistically Interpreted Corpora (LINC-03), *EACL 2003*, pages 109–116.

Przepiórkowski, A., Kupść, A., Marciniak, M., and Mykowiecka, A. (2002). *Formalny opis języka polskiego: Teoria i implementacja*. Akademicka Oficyna

Wydawnicza EXIT, Warsaw.

Przepiórkowski, A., Bański, P., Łukasz Dębowski, Hajnicz, E., and Woliński, M. (2003). Konstrukcja korpusu IPI PAN. *Polonica*, **XXII–XXIII**, 33–38.

Przepiórkowski, A., Hajnicz, E., Woliński, M., and Dębowski, Ł. (2004a). Zasady znakowania morfosyntaktycznego w Korpusie IPI PAN. Ms., Institute of Computer Science, Polish Academy of Sciences.

Przepiórkowski, A., Krynicki, Z., Dębowski, u., Woliński, M., Janus, D., and Bański, P. (2004b). A search tool for corpora with positional tagsets and ambiguities. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004*, pages 1235–1238.

Saloni, Z. (1976). Kategoria rodzaju we współczesnym języku polskim. In *Kategorie gramatyczne grup imiennych we współczesnym języku polskim*, pages 41–75. Ossolineum, Wrocław.

Saloni, Z. (1977). Kategorie gramatyczne liczebników we współczesnym języku polskim. *Studia Gramatyczne*, **I**, 145–173.

Saloni, Z. (1981). Uwagi o opisie fleksyjnym tzw. zaimków rzeczownych. *Folia Linguistica*, **2**, 265–271.

Saloni, Z. (1988). O tzw. formach nieosobowych [rzeczowników] męskoosobowych we współczesnej polszczyźnie. *Biuletyn Polskiego Towarzystwa Językoznawczego*, **XLI**, 155–166.

Saloni, Z. (2001). *Czasownik polski. Odmiana, słownik*. Wiedza Powszechna, Warsaw.

Tokarski, J. (1993). *Schematyczny indeks* a tergo *polskich form wyrazowych*. Wydawnictwo Naukowe PWN, Warsaw. Elaborated and edited by Zygmunt Saloni.

Woliński, M. (2001). Rodzajów w polszczyźnie jest osiem. In W. Gruszczyński, U. Andrejewicz, M. Bańko, and D. Kopcińska, editors, *Nie bez znaczenia... Prace ofiarowane Profesorowi Zygmuntowi Saloniemu z okazji jubileuszu 15000 dni pracy naukowej*, pages 303–305. Wydawnictwo Uniwersytetu Białostockiego, Białystok.

Woliński, M. (2003). System znaczników morfosyntaktycznych w korpusie IPI PAN. *Polonica*, **XXII–XXIII**, 39–55.

Woliński, M. and Przepiórkowski, A. (2001). Projekt anotacji morfosynktaktycznej korpusu języka polskiego. IPI PAN Research Report 938, Institute of Computer Science, Polish Academy of Sciences.

Index