Web Service integration platform for Polish linguistic resources

Maciej Ogrodniczuk, Michał Lenart

Institute of Computer Science
Polish Academy of Sciences
ul. Ordona 21, Warsaw, Poland
maciej.ogrodniczuk@ipipan.waw.pl, michal.lenart@gmail.com

Abstract

This paper presents a robust linguistic Web service framework for Polish, combining several mature offline linguistic tools in a common online platform. The toolset comprise paragraph-, sentence- and token-level segmenter, morphological analyser, disambiguating tagger, shallow and deep parser, named entity recognizer and coreference resolver. Uniform access to processing results is provided by means of a stand-off packaged adaptation of National Corpus of Polish TEI P5-based representation and interchange format.

A concept of asynchronous handling of requests sent to the implemented Web service (Multiservice) is introduced to enable processing large amounts of text by setting up language processing chains of desired complexity. Apart from a dedicated API, a simple Web interface to the service is presented, allowing to compose a chain of annotation services, run it and periodically check for execution results, made available as plain XML or in a simple visualization. Usage examples and results from performance and scalability tests are also included.

Keywords: Web services, language processing chains, Polish, TEI P5, CLARIN

1. Introduction

Following the CLARIN[1] perspective of making language processing tools available online, approximately 200 Web services for 20 languages have been made available (Ogrodniczuk and Przepiórkowski, 2010; Ogrodniczuk and Przepiórkowski, 2011) in CLARIN preparatory phase, finished in June 2011.

This apparent abundance resulted in several initiatives aiming at creating the general technological frameworks for combining these processing tools into ready-to-use packages by offering chaining possibilities. This had to be followed by reuse or adaptation of common representation format for the chain, general enough to cover the diversity of linguistic description on multiple levels.

One of such platforms, further referred to as *the Multiservice* was created by the Linguistic Engineering Group at the Institute of Computer Science, Polish Academy of Sciences (ICS PAS). It consists of the common TEI P5-based representation format, asynchronous execution architecture and several interfaces for Web service chaining, including a Web-based application.

2. The Integrated Tools

Offline versions of all integrated tools have been used by the linguistic community in Poland for several years and they proved their suitability and efficiency for linguistic engineering tasks. They constitute the basic building blocks of many local processing chains, but have never been made available online in a consistent manner[2] (in a common infrastructure and format). In this respect, bringing the tools online can go beyond just illustrating capabilities of the framework and the format and constitute real value to the community and other interested parties. All integrated tools are open source and all are actively maintained and developed.

Here is the short list of Polish processing tools currently available in the framework:

- Morfeusz (Woliński, 2006b) is a morphological analyzer for Polish using a positional tagset (Przepiórkowski and Woliński, 2003); current version of the tool, Morfeusz SGJP, is based on linguistic data coming from The Grammatical Dictionary of Polish (Saloni et al., 2007) – see also `http://sgjp.pl/morfeusz.html`,
- TaKIPI (Piasecki and Wardyński, 2006) is a hybrid (multiclassifier) rule-based morphosyntactic tagger (disambiguator of morphological descriptions) of Polish,
- Pantera (Acedański, 2010) is a recently developed morphosyntactic rule-based Brill tagger of Polish using an optimized version of Brill's algorithm adapted for specifics of inflectional languages,
- Spejd (Buczyński and Przepiórkowski, 2009) is an engine for shallow parsing using cascade regular grammars with their own specification language supporting accepting and rejecting morphological interpretations, agreement of entire tags or particular grammatical categories as well as grouping (with syntactic and semantic head specified independently),
- Świgra (Woliński, 2006a) is a Prolog deep parser of Polish implementing Marek Świdziński's formal metamorphosis grammar of Polish (Świdziński, 1992), regarded as the largest and most precise formal description of general grammar of Polish,

[1]Common Language Resources and Technology Infrastructure, see e.g. `http://www.clarin.eu/`.

[2]Demo online version of the older variant of the morphological analyser, Morfeusz SiAT, is still available at `http://sgjp.pl/demo/morfeusz`; TaKIPI tagger is also wrapped as a separate Web service which can be tested at `http://nlp.pwr.wroc.pl/clarin/ws/takipi/`.

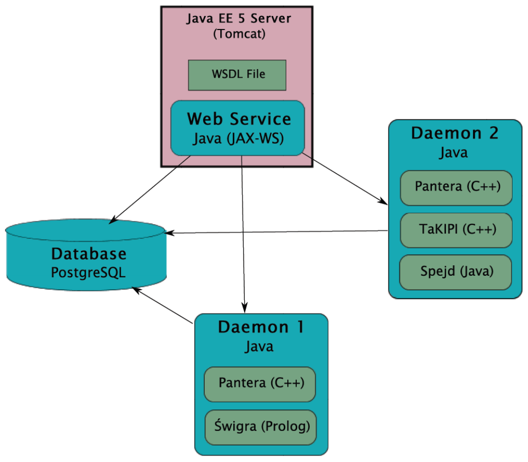


Figure 1: Architecture of the Multiservice

- NERF (Savary et al., 2010) is a statistical CRF-based named entity recognizer trained over 1-million-subcorpus of National Corpus of Polish (Przepiórkowski et al., 2010) and successfully used in the process of automated annotation of its total 1 billion segments,
- Ruler (Ogrodniczuk and Kopeć, 2011) is a rule-based end-to-end coreference resolution system for Polish using syntactic constraints (e.g. elimination of nested nominal groups), syntactic filters (e.g. elimination of syntactic incompatible heads) and selection (weighted scoring) for identity-of-reference chain detection.

3. The Architecture of the Online Service

Language processing chains are created by sending requests to the Web service which are being handled in asynchronous manner to allow processing large amounts of text. Invoking one of the available methods results in returning the request token (identifier) which can be used to check the request status and retrieve the result when processing completes. This design is directly inspired by TaKIPI Web service (Broda et al., 2010) prepared by ICS PAS and Wrocław University of Technology.

Requests are first enqueued (by executing `analyzeChain(text, chainParts, inputFormat, outputFormat)` method) to let the service execute the given chain of operations on the given text. Each part of the chain is defined by operation type (i.e. linguistic function such as tagging or shallow parsing), requested tool name (since there can be many variant tools of the same type configured) and a map of properties specific to the provided tool. By using chains, one request can trigger several (interrelated or independent) operations at once, e.g. "tag text with Pantera, then perform deep parsing with Świgra, filtering results using disambiguation information provided by Pantera").

The chain registering function returns a unique token of the request which can be then queried for (by means of `getStatus(token)` method) to periodically check the status of request with the given token (currently one of: `PENDING`, `IN_PROGRESS`, `DONE` or `FAILED`). If the status of request is `DONE`, `getResult(token)` function can be executed to return the result of the chain execution (an XML TEI P5 file, see Sec. 4.). The same function is used to return an error message when execution `FAILED`. Removal of requests is currently not possible.

4. The Packaged TEI P5-based Linguistic Representation

To achieve chaining linguistic tools, a common representation and interchange format was necessary to bind the tools together and present results to the user. Such format has been developed for the National Corpus of Polish as a stand-off, TEI P5-encoded annotation which stores different levels of description in separate, interlinked files (Bański and Przepiórkowski, 2009; Bański and Przepiórkowski, 2010; Przepiórkowski and Bański, 2009) similarly to PAULA (Dipper, 2005) or MAF (Clement and de la Clergerie, 2005). For the current interchange activity, a certain method of packaging them has been applied, keeping the NKJP annotation principles on one hand and adapting it to uniformity and performance requirements of the online service on the other.

To meet packaging requirements, the standard TEI approach to storing multiple files in a single entity with creation of an (artificial) corpus has been used, adopting the following structure:

- root `<teiCorpus>` element represents the collection of annotation layers (including the source text),
- child `<TEI>` elements represent each annotation layer (e.g. segmentation, lemmatization etc.)

Linguistic annotations of different levels are stored as `<text>`s of the corpus with embedded `<body>` and `<p>` subelements from the TEI syntax model. At each level minimalistic set of means of TEI expression was used to preserve the information: neutral `<seg>`ments linked by means of `corresp` attributes and TEI-embedded feature structure mechanism preserving structured annotation.

Currently the following layers have been integrated:

- text structure layer: contains "original" representation of text, paragraph-segmented,
- segmentation layer: contains division of text into paragraphs, sentences and tokens,
- morphosyntactic layer: contains disambiguated lexical interpretations of referenced tokens – POS and morphosyntactic tags – together with identified lemmata (with alternative values grouped in `vAlt` elements),
- syntactic word layer: groups individual tokens into higher-level units to facilitate parsing,
- syntactic group (shallow parsing) layer: represents syntactic groups with pointers (`<ptr>`s) to immediate constituents of the group — syntactic words or other syntactic groups, with syntactic and semantic heads specifically marked,

- named entity layer: represents hierarchical named entities providing information on their types,
- deep parsing layer: represents parsing results as a shared parse forest, i.e. a collection of parse subtrees stored in a packed graph format, with each unique subtree stored only once,
- coreference layer: provides information about mentions and coreferential chains.

Below we present a sample partial description from the morphosyntactic layer (see `http://nlp.ipipan.waw.pl/TEI4NKJP/` for more detailed and complete examples):

```
<seg xml:id="m-seg1" corresp="#s-seg1">
 <fs type="morph">
  <f name="interps">
   <vAlt>
    <fs type="lex"
        xml:id="m-seg1-lex">
     <f name="base">
       <string>lato</string></f>
     <f name="ctag">
       <symbol value="subst"/></f>
     <f name="msd">
       <symbol value="pl:gen:n"
               xml:id="m-seg_1-msd"/>
    </fs>
    <fs type="lex"
        xml:id="m-seg2-lex">
     <f name="base">
       <string>rok</string></f>
     <f name="ctag">
       <symbol value="subst"/></f>
     <f name="msd">
       <symbol value="pl:gen:m3"
               xml:id="m-seg_2-msd"/>
    </fs>
   </vAlt>
  </f>
  <f name="disamb">
   <fs feats="#pantera"
       type="tool_report">
    <f fVal="#m-seg_2-msd"
       name="choice"/>
    <f name="interpretation">
     <string>rok:subst:pl:
            gen:m3</string>
     ...
```

5. Interface and Usage

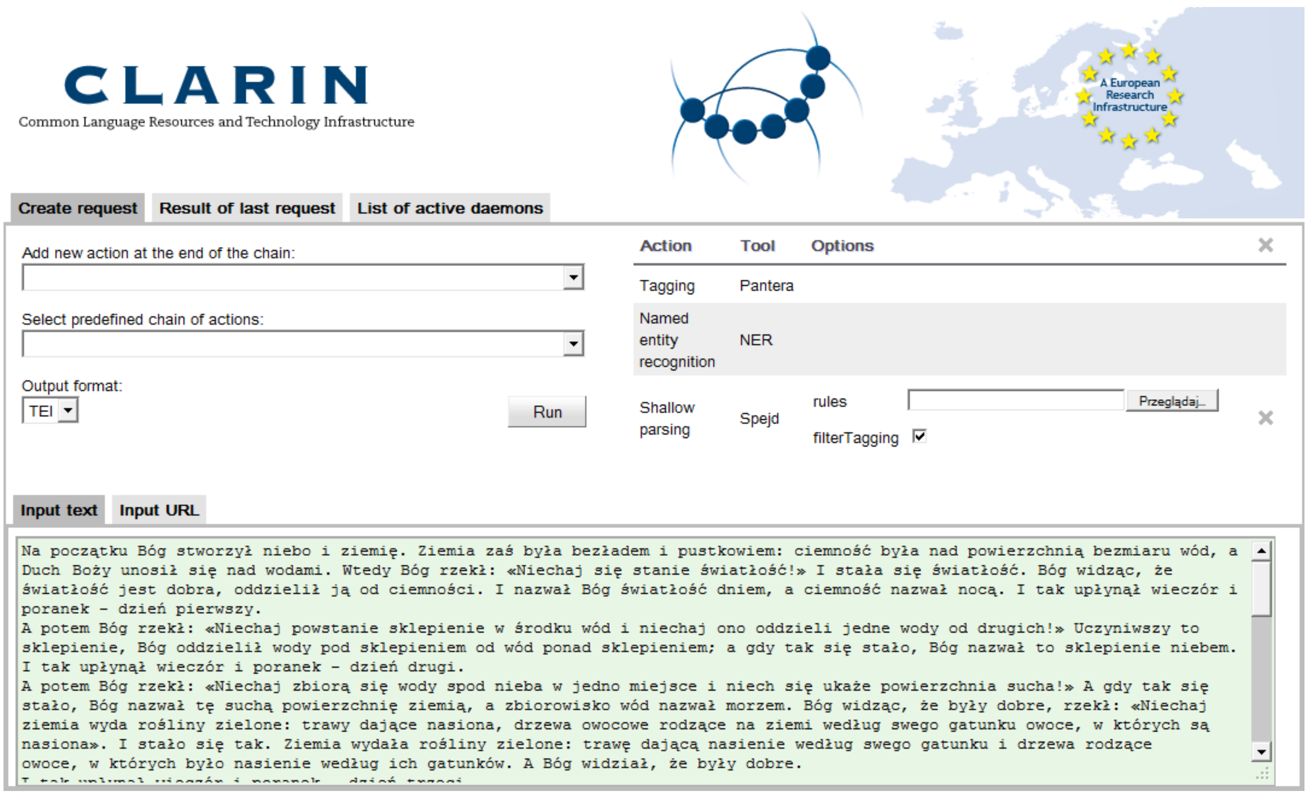
The service is executed in the following manner:

- the user sends to the service a processing request with the linguistic function name and its parameters,
- the service generates a token for the request (further used to operate on the given request with the service), stores the request in the queue and returns the token to the user,
- the user keeps querying the service about the status of execution of a request identified with a given token until the status shows that the execution stopped because of error or ended successfully,
- on information on execution success (or failure), the result (or an error message) can be retrieved by the user and execution stops.

The *pull* execution method gives potential interfaces far more flexibility and allows for better control over annotation processes as compared to callback-based implementations. This is especially important with respect to the recent hype of using Web services for processing corpora which may result in long processing times.
The Multiservice is intended to be used via a dedicated API, but to offer online access, a simple Web interface (see Fig. 2) is available at `http://chopin.ipipan.waw.pl/multiservice/`. It allows to enter the text (or URL containing it) to be processed and compose a chain of annotation services.
After starting the analysis, the Web application checks periodically the status of the request. When it ended execution, the result is retrieved and displayed to the user. In case of a failure, an appropriate error message is presented.

6. Notes on Performance and Scalability

Service plugin implementation class does not require the tool being integrated to run as a background daemon (in comparison to starting it for each request separately). However, all existing tools are initialized only once – at the daemon startup. Intermediate results are parsed using StAX-based parser, without keeping the whole XML document in memory. Therefore it is possible to handle quite large text documents. Nevertheless XML parsing still creates significant overhead.
The major challenge for linguistic services is also storing potentially huge amounts of data resulting from processing of rather small input. It is not uncommon e.g. for deep parsing systems which could generate plenty of different result trees even though their subtrees are identical among subsequent results. In case of the Multiservice the problem was diminished by using a shared parse forest representation with each unique subtree stored no more than in one instance.
A simple performance test has been carried out for one of the component services. It shows overhead caused by using the Web service when performing a simple request of tagging n-word text with Pantera tagger (against the execution of the offline tool alone). Results in Table 1 show durations of request execution through Web service on the local host for the most time-consuming operations.
It clearly shows that total request execution is 2-3 times longer than invoking the tagger locally. Even though for shorter texts such as a typical newspaper article (of approx. 1 000 tokens) the difference can still seem acceptable, various optimizations are considered. Improvements in XML postprocessing (inclusion of headers, pretty-printing of document generated by C++ application with Pantera) should give the most significant performance boost.


CLARIN
Common Language Resources and Technology Infrastructure
A European Research Infrastructure
Create request
Result of last request
List of active daemons
Add new action at the end of the chain:
Select predefined chain of actions:
Run
Przeglądaj...
Input text
Input URL

Output format: TEI

Action	Tool	Options
Tagging	Pantera	
Named entity recognition	NER	
Shallow parsing	Spejd	rules; filterTagging ✓

```
Na początku Bóg stworzył niebo i ziemię. Ziemia zaś była bezładem i pustkowiem: ciemność była nad powierzchnią bezmiaru wód, a
Duch Boży unosił się nad wodami. Wtedy Bóg rzekł: «Niechaj się stanie światłość!» I stała się światłość. Bóg widząc, że
światłość jest dobra, oddzielił ją od ciemności. I nazwał Bóg światłość dniem, a ciemność nazwał nocą. I tak upłynął wieczór i
poranek - dzień pierwszy.
A potem Bóg rzekł: «Niechaj powstanie sklepienie w środku wód i niechaj ono oddzieli jedne wody od drugich!» Uczyniwszy to
sklepienie, Bóg oddzielił wody pod sklepieniem od wód ponad sklepieniem; a gdy tak się stało, Bóg nazwał to sklepienie niebem.
I tak upłynął wieczór i poranek - dzień drugi.
A potem Bóg rzekł: «Niechaj zbiorą się wody spod nieba w jedno miejsce i niech się ukaże powierzchnia sucha!» A gdy tak się
stało, Bóg nazwał tę suchą powierzchnię ziemią, a zbiorowisko wód nazwał morzem. Bóg widząc, że były dobre, rzekł: «Niechaj
ziemia wyda rośliny zielone: trawy dające nasiona, drzewa owocowe rodzące na ziemi według swego gatunku owoce, w których są
nasiona». I stało się tak. Ziemia wydała rośliny zielone: trawę dającą nasienie według swego gatunku i drzewa rodzące
owoce, w których było nasienie według ich gatunków. A Bóg widział, że były dobre.
```

Figure 2: The Multiservice Web interface

Table 1: Tagging performance test results

Number of words	Execution time (s)			
	Total	Tagger	XML	DB
1 000	3.2	1.0	1.2	0.1
2 000	4.5	1.4	2.1	0.6
5 000	9.2	3.7	3.3	1.4
10 000	14.9	7.0	4.8	2.7
20 000	32.6	14.7	10.3	5.7
30 000	48.3	22.4	15.7	8.1
40 000	98.5	29.7	45.3	22.1

It should also be noted that a test version of Multiservice using binary data based on Apache Thrift library as interchange format is currently under development. It is expected to give even better performance results as XML document will be created only at the very end of the execution process.

7. Conclusions and Further Work

Leaving aside the purely engineering task of providing advanced linguistic services for Polish in a common framework, preparation of the Multiservice resulted in several useful developments. The first of them is by far the common tagset-independent TEI P5-based format, proved effective in an undergoing coreference resolution experiments where it serves as a representation and interchange format for pre-resolution linguistic analysis. Being standoff and packaged at the same time and basing on the idea of a TEI corpus, the format is flexible enough to create bundles, represent layers and annotation variants separately which facilitates comparisons of different annotation models (e.g. two sets of parsing trees produced for two different results of tagging over the same input). Another important aspect of the realized solution is definitely the asynchronous character of the service, making it ideal for processing large amounts of data in a convenient way.

The first practical outcome of the implementation is its application to the recent attempts of coreference resolution for Polish where the Multiservice is used to tokenize, disambiguate morphological description and detect noun phrases in the analyzed text.

Further technical work would concentrate on integration of a wider range of input and output formats with multiple encodings and integrated converters as well as plugging in existing offline annotation components[3] and future higher-level annotation tools such as word-sense disambiguators.

Research activities would further delve into aspects of maintaining semantic interoperability of the newly developed format (including issues related to Polish morphosyntax vs. e.g. ISOCat Data Category Registry) as well as Web service chaining issues such as algorithms for automated chain detection, currently offered by WebLicht (Hinrichs et al., 2010).

[3]First candidates are competitive morphological analysers such as Morfologik by Marcin Miłkowski (see `http://morfologik.blogspot.com/`) or complete language processing toolsets such as UAM Text Tools (see `http://atos.wmid.amu.edu.pl/~obrebski/ptx/utt/utt.html`).

8. Acknowledgements

The work reported here was carried out within the Common Language Resources and Technology Infrastructure (CLARIN) project co-funded by the European Commission under the Seventh Framework Programme – Capacities Specific Programme Research Infrastructures (Grant Agreement No 212230).

9. References

Szymon Acedański. 2010. A Morphosyntactic Brill Tagger for Inflectional Languages. In Hrafn Loftsson, Eiríkur Rögnvaldsson, and Sigrún Helgadóttir, editors, *Advances in Natural Language Processing*, volume 6233 of *Lecture Notes in Computer Science*, pages 3–14. Springer.

Piotr Bański and Adam Przepiórkowski. 2009. Stand-off TEI Annotation: the Case of the National Corpus of Polish. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III) at ACL-IJCNLP 2009*, pages 64–67, Singapore.

Piotr Bański and Adam Przepiórkowski. 2010. The TEI and the NCP: the model and its application. In *LREC 2010 Workshop on Language Resources: From Storyboard to Sustainability and LR Lifecycle Management*, Valletta, Malta. ELRA.

Bartosz Broda, Michał Marcińczuk, and Maciej Piasecki. 2010. Building a node of the accessible language technology infrastructure. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, Mike Rosner, Daniel Tapias (eds.), May 19-21*, Valletta, Malta.

Aleksander Buczyński and Adam Przepiórkowski. 2009. Spejd: A shallow processing and morphological disambiguation tool. In Zygmunt Vetulani and Hans Uszkoreit, editors, *Human Language Technology: Challenges of the Information Society*, volume 5603 of *Lecture Notes in Artificial Intelligence*, pages 131–141. Springer-Verlag, Berlin.

Lionel Clement and Eric Villemonte de la Clergerie. 2005. MAF: a morphosyntactic annotation framework. In *Proceedings of the 2nd Language & Technology Conference*, pages 90–94, Poznań.

Stefanie Dipper. 2005. XML-based Stand-off Representation and Exploitation of Multi-Level Linguistic Annotation. In *Proceedings of Berliner XML Tage 2005 (BXML 2005)*, pages 39–50, Berlin.

Marie Hinrichs, Thomas Zastrow, and Erhard Hinrichs. 2010. Weblicht: Web-based LRT Services in a Distributed eScience Infrastructure. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC 2010*, Valletta, Malta. ELRA.

Maciej Ogrodniczuk and Mateusz Kopeć. 2011. End-to-end coreference resolution baseline system for Polish. In Zygmunt Vetulani, editor, *Proceedings of the 5th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 167–171, Poznań, Poland.

Maciej Ogrodniczuk and Adam Przepiórkowski. 2010. Linguistic Processing Chains as Web Services: Initial Linguistic Considerations. In *Proceedings of the Workshop on Web Services and Processing Pipelines in HLT: Tool Evaluation, LR Production and Validation (WSPP 2010) at the Language Resources and Evaluation Conference (LREC 2010)*, pages 1–7, Valletta, Malta. ELRA.

Maciej Ogrodniczuk and Adam Przepiórkowski. 2011. Integration of Language Resources into Web service infrastructure. Technical report. CLARIN deliverable D5R-3b.

Maciej Piasecki and Adam Wardyński. 2006. Multiclassifier Approach to Tagging of Polish. In *Proceedings of 1st International Symposium Advances in Artificial Intelligence and Applications*.

Adam Przepiórkowski and Piotr Bański. 2009. XML Text Interchange Format in the National Corpus of Polish. In Stanisław Goźdź-Roszkowski, editor, *The proceedings of Practical Applications in Language and Computers PALC 2009*, Frankfurt am Main. Peter Lang. Forthcoming.

Adam Przepiórkowski and Marcin Woliński. 2003. A Flexemic Tagset for Polish. In *Proceedings of* Morphological Processing of Slavic Languages*, EACL 2003*.

Adam Przepiórkowski, Rafał L. Górski, Marek Łaziński, and Piotr Pęzik. 2010. Recent developments in the National Corpus of Polish. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC 2010*, Valletta, Malta. ELRA.

Zygmunt Saloni, Włodzimierz Gruszczyński, Marcin Woliński, and Robert Wołosz. 2007. *Słownik gramatyczny języka polskiego*. Wiedza Powszechna, Warsaw. 177 pp., CD.

Agata Savary, Jakub Waszczuk, and Adam Przepiórkowski. 2010. Towards the annotation of named entities in the National Corpus of Polish. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC 2010*, Valletta, Malta. ELRA.

Marek Świdziński. 1992. *Formal grammar of Polish.* [In Polish]. Warsaw University Dissertations, Warsaw.

Marcin Woliński. 2006a. Jak się nie zgubić w lesie, czyli o wynikach analizy składniowej według gramatyki Świdzińskiego. *Poradnik Językowy*, 9:102–114.

Marcin Woliński. 2006b. Morfeusz — a practical tool for the morphological analysis of Polish. In Mieczysław A. Kłopotek, Sławomir T. Wierzchoń, and Krzysztof Trojanowski, editors, *Proceedings of the International Intelligent Information Systems: Intelligent Information Processing and Web Mining'06 Conference*, pages 511–520, Wisła, Poland, June.