Maca — a configurable tool to integrate Polish morphological data*

Adam Radziszewski
Institute of Informatics
Wrocław University of Technology
Wybrzeże Wyspiańskiego 27,
Wrocław, Poland
adam.radziszewski@pwr.wroc.pl

Tomasz Śniatowski
Institute of Informatics
Wrocław University of Technology
Wybrzeże Wyspiańskiego 27,
Wrocław, Poland
kailoran@gmail.com

Abstract

There are a number of morphological analysers for Polish. Most of these, however, are non-free resources. What is more, different analysers employ different tagsets and tokenisation strategies. This situation calls for a simple and universal framework to join different sources of morphological information, including the existing resources as well as user-provided dictionaries. We present such a configurable framework that allows to write simple configuration files that define tokenisation strategies and the behaviour of morphological analysers, including simple tagset conversion.

1 Introduction

In this paper we will focus on two tasks of Natural Language Processing: *tokenisation* and *morphological analysis*. Both tasks usually precede the application of a morpho-syntactic tagger and/or some type of a parser. Tokenisation is the process of dividing running text into parts corresponding to words and word-like units (Grefenstette and Tapanainen, 1994). Morphological analysis consists of assigning a *morphological description* to each of these tokens. Morphological analysis does not take into account context, i.e. we perform a dictionary look-up without accounting for the actual role of the token attributed by the context of usage. These two tasks are tightly coupled: the characteristics of the analysed language, anticipated usage scenario and preferred grammatical theory will influence the choice of how narrow a stretch of text we will want to isolate and then, account for by assigning a morphological description.

The employed types of morphological description may also differ. Typically this is a set of *lemma–tag* pairs assigned to each token, where *lemma* is the dictionary form of the token and *tag* is a morpho-syntactic tag describing the grammatical class and possibly some inflectional and syntactic properties. Throughout this paper we will be using this interpretation. For instance, the morphological dictionary of *Morfologik 1.6* (Miłkowski, 2010) contains the following entries for the Polish form *maca* (the second column contains lemmas, the third one — tags):

```
maca  mac    subst:irreg
maca  maca   subst:sg:nom:f
maca  macać  verb:fin:sg:ter:imperf
```

There are two noun (`subst`) readings: the first one, seemingly erroneous, could refer to a Macintosh computer (*mac*) in the genitive case (the proper tag would be `subst:sg:f:m2`), the other one standing for the noun *maca* (*matzo*, *unleavened bread*) in nominative case, singular number. The last entry corresponds to a finite verb *macać* (*to palpate, to paw*) in third person, singular number and imperfective aspect. Having taken a sentence with this token, we would expect a morpho-syntactic tagger to select only one of these descriptions as contextually appropriate (depending on the whole sentence).

Acknowledgement. This work is financed by Innovative Economy Programme project POIG.01.01.02-14-013/09.

F. Sánchez-Martínez, J.A. Pérez-Ortiz (eds.)
Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation, p. 29–36
Barcelona, Spain, January 2011. http://hdl.handle.net/10609/5645

Several morphological analysers have been created for Polish (Hajnicz and Kupść, 2001; Woliński, 2006) but, to the best of our knowledge, only one of them has been released under a free licence (as of November 2010): *Morfologik*, a part of the *LanguageTool* open source proofreading tool (Miłkowski, 2010). Morfologik per se is not a piece of software but rather a single tab-separated text file containing a morphological dictionary. The dictionary is quite sizeable (Morfologik 1.6 contains entries for nearly 3.5 million forms) and is available under a dual licence: GNU LGPL or Creative Commons Share Alike (the user is free to choose).

Other analysers are at best free for scientific or non-commercial purposes[1]. We will mention two of them:

1. *Morfeusz SIaT* (Woliński, 2006), a dynamic library with an embedded dictionary released under a restrictive licence (the dictionary is not a separate data file and attempts at extracting the data are explicitly forbidden by the licence). Most importantly, this is the only analyser compatible with the free manually annotated corpus of Polish — a part of the IPI PAN Corpus of Polish (Przepiórkowski, 2004)[2]. This compatibility lies in using the same tagset and sticking to the same guidelines of tag usage. Another advantage of Morfeusz is that its data is a result of many years of work by recognised Polish linguists. It is worth pointing out that Morfeusz is not a typical morphological dictionary: instead of the usual token-by-token analysis, it outputs graph structures that can account for segmentation ambiguities.

2. *Polex/PMDBF* morphological dictionary (Vetulani, 2000) distributed with *UAM Text Tools* package (Obrębski and Stolarski, 2006). The dictionary contains about 1 million forms and is licensed under Creative Commons Attribution Non-Commercial Share Alike (CC BY-NC-SA). The main advantage of the analyser is the format of its data, a plain text file.

The aforementioned manually annotated corpus is a precious resource, since it is the only free data source that may be used to train a morpho-syntactic tagger for Polish. The corpus has been released under GNU GPL and contains 660 000 tokens. Although it can be used as a material to extract morphological data, it is not large enough to get a reasonable coverage for less frequent forms. To amend this, external morphological analysers must be used. A preferred solution would assume combining available data sources in a way that would allow:

- configuration of the multi-analyser system,
- selecting which analysers to use at the moment (e.g. being able to use Morfeusz but with the possibility of not requiring it when enough free data is available),
- setting up an analysis pipeline (when one analyser fails to produce an analysis, another might),
- overriding erroneous entries in existing dictionaries (analysers),
- accounting for some differences in tagsets and tokenisation strategies,
- tying different analysis pipelines to different token types (e.g. some characters will anyway be recognised as punctuation during tokenisation, why not use that information),
- handling large dictionaries with reasonable memory load and processing efficiency.

In the rest of this paper we discuss some important details of the mentioned Polish language resources and then present our solution that meets the above requirements: the Maca system (*Morphological Analysis Converter and Aggregator*).

2 Tagsets and tag representation

Each of the mentioned analysers has its own tagset. The tagset of Morfeusz (and the whole IPI

[1] Non-commercial licences are not really *free*, since they forbid quite a number of basic usage scenarios, including on-line advertising on a site that uses the licensed material. What is more, they are incompatible with many *free* licences, including the GNU GPL.

[2] This corpus is in fact a re-annotated version of the *corpus of the Frequency dictionary of contemporary Polish* (Ogrodniczuk, 2003). The corpus can be downloaded from `http://korpus.pl/index.php?page=download`.

PAN Corpus, henceforth *IPIC tagset*) has been designed with specific criteria in mind, most notably treating inflection as primary means of distinguishing grammatical classes (Przepiórkowski and Woliński, 2003). This results in having a rather fine-grained decomposition in comparison with traditional parts-of-speech. What is more, the tagset is *positional* — i.e. each tag contains a grammatical class and zero or more *values* for some *attributes*. Each grammatical class defines a set of attributes whose values must be specified. The attributes with their values, and grammatical classes with valid/required attributes define a tagset. For instance, nouns require that number, gender and case attributes are specified, and adverbs require the degree attribute. Forms traditionally classified as adverbs but non-gradable and not derived from adjectives are treated as *particle-adverbs*. The only exception to this rigour is that some less important attributes are optional for some grammatical classes, e.g. prepositions require that case is specified, but may also contain the *vocalicity* attribute[3]. Technically, these optional attributes must come after the required attributes.

Tags in the IPIC tagset are represented as simple colon-separated strings always starting with grammatical class. Dot and underscore characters are used to represent multiple tags in compact format — e.g. `subst:_:nom.acc:f` stand for four tags: `subst:sg:nom:f`, `subst:sg:acc:f`, `subst:pl:nom:f`, `subst:pl:acc:f` (underscore stands for all possible values of the attribute in this position — in case of the `subst` class, number — while the dot character separates possible values of one attribute, in this instance the case attribute). Note that this is merely a notation shorthand.

The tags appearing in the morphological dictionary of Morfologik closely resemble the IPIC tagset. According to the *Readme* file, this is intended. Unfortunately, the tagset as such is hardly defined. The *Readme* file enumerates grammatical classes and values of some (unnamed) attributes but it does not state which values apply to which classes. In fact, it cannot be precisely defined, as for most of the classes there exist several variants, practically rendering most of the attributes optional (cf. the example in the previous section — some nouns are claimed to be "irregular" without providing any other details; the IPIC tagset would require such forms to constitute a separate grammatical class). The tags are represented in the same textual format as in Morfeusz; furthermore, many classes are described in almost the same manner. Thus, by means of semi-automatic conversion, a huge part of the morphological dictionary could be converted into the IPIC tagset.

The tagset of Polex/PMDBF is quite traditional. This is reflected by the choice of grammatical classes quite literally being traditional parts-of-speech. The assignment of attributes to grammatical classes, although not explicitly stated in the manual, can easily be inferred from actual tags. Textual representation is helpful in this respect, as value abbreviations are prefixed with attribute mnemonics — e.g. `N/GfNsCn` stands for noun in feminine gender, singular number and nominative case.

3 Segmentation issues

The IPIC tagset assumes quite a specific tokenisation strategy. A strict rule is that a token cannot contain any whitespace. On the other hand, some forms are divided into several tokens. This happens in case of so-called *floating inflections*, which are treated as forms of the verb *być* (*to be*) (Woliński, 2006). For instance, *czytałem* (*I was reading*) is split into *czytał* (l-participle in singular masculine) and *em* (in first person, singular number). Similarly, *czytałbym* (the same form in conjunctive mood) is split into *czytał*, *by* (conjunctive particle) and *m*. Note that this splitting is performed on the level of surface forms and each of the isolated tokens is assigned a set of lemma–tag pairs.

In some cases one form may be analysed in different ways, each calling for different tokenisation. Such cases are the reason why Morfeusz always outputs directed acyclic graphs (DAGs) Woliński (2006) instead of a plain sequence of to-

[3] *Vocalicity* accounts for some phonological variations of the word form. In the case of prepositions, the variation is manifested in in the vowel *e* added as a suffix when the preposition occurs before certain consonants; e.g. *nad nami* (*over us*) v. *nade mną* (*over me*). Tags for prepositions that are not affected by the phenomenon, e.g. *na* (*on*), contain no value of the attribute.

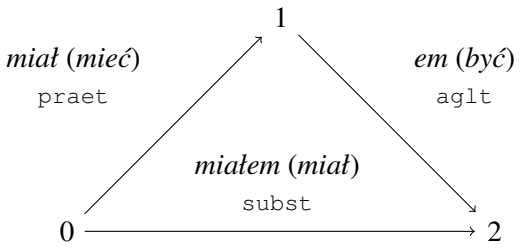

Figure 1: Morfeusz output for the form *miałem*: the upper path having a verb reading (*I had*), the lower one — a noun (*dust*) in instrumental case, singular. Lemmas are given in parentheses, tags are abbreviated.

kens. For instance, the form *miałem* will be analysed as presented on Fig. 1. The verb interpretation is split into two tokens, while the other one is a single token marked as a noun.

This behaviour of Morfeusz is actually both a blessing and a curse: while getting the most accurate description of the surface phenomena, one cannot directly use many standard NLP techniques where input is assumed to be sequential. For instance, virtually all morpho-syntactic tagging algorithms assume that the input is represented as a linear sequence of tokens, not a fancy graph structure. What is more, the two other analysers use a more traditional tokenisation strategy, which renders automatic conversion of verbs non-trivial. This is also a reason why it was impossibly to incorporate Morfeusz directly into an existing framework for morphological analysis.

4 System architecture

Maca was devised as a way of combining morphological data from various, not trivially compatible sources. The primary goals of the technical side of the project were to have the whole functionality available as both a command-line tool and a C or C++ library for use in other NLP software such as taggers and parsers. The choice of language was dictated by the need for performance, reasonable abstraction level and relative ease of interoperation with C++ and Python. The entire project was split into several libraries, partly due to the desire to re-use some parts in other projects without introducing one large dependency.

The project has been released under GNU GPL 3.0 and may be obtained from `http://nlp.pwr.wroc.pl/redmine/projects/libpltagger`.

4.1 Toki

The tokeniser, Toki, resides in a separate library and does not deal with morphology at all, instead offering a layered architecture for splitting tokens according to configurable rules. More than just a plain whitespace tokenisation is needed due to issues such as punctuation, that we would prefer to be split into separate tokens, and punctuation-rich text entities that we would prefer not to split, such as URLs. The rules, stored in a text (INI-like) format which can be easily modified, allow a reasonably quick way of trying out various approaches like lists of affixes or regular expressions. Tokens can be tagged with a *token type*, using a dictionary or, again, a regular expression. These type labels are useful both internally, allowing conditional layer application, and externally, allowing some initial token classification.

It is currently assumed that whitespace characters are always token separators. These characters are discarded and each token is assigned a qualitative desciption of the type and amount of whitespace that preceded it. Note that this is the only information loss that happens to the input text — this way it is guaranteed that a tokenised input may be reverted to the original text with only minor changes in blanks. What follows is that neither Toki, nor Maca deal with multiword units. The motivation for this decision was twofold: 1) multiword unit recognition entails resolving ambiguities, which seems too difficult for this processing step and 2) this is compliant with the segmentation policy coupled with the IPIC tagset (Przepiórkowski and Woliński, 2003). Multiword units can be accounted for on higher levels of processing, e.g. a chunker or Named Entity tagger could be used to group together expressions such as *San Francisco*. Note that, however, this behaviour could be altered by enhancing Toki with a simple token joining layer, or by enhancing Toki with a simple token joining layer or by modyfying the initial tokenisation module.

Additionally, Toki can perform sentence splitting using SRX (*Segmentation Rules eXchange*)

rules (Pooley and Raya, 2008). SRX is a standard for segment boundary detection, based on regular expressions, with the important advantage that there exist freely available sentence segmentation rules for many languages, including a LGPL-licensed set of rules for Polish (Miłkowski and Lipski, 2009). SRX needs to be run on plain, not tokenised text, and therefore has been implemented as an (optional) first processing level in Toki. Incidentally, the Toki SRX support seems to be the first freely available C++ implementation of the SRX standard, as all the libraries we could find were in either Java or Python. We decided on a new implementation in order to avoid having to call a high-level, VM-based library at such a low level in our processing pipeline — performance seems to be on par with the Java SRX implementations we tested (Miłkowski and Lipski, 2009).

Toki also contains a simple utility for testing rules called `toki-app` exposing most of the library's features, including the SRX segmentation.

4.2 Corpus2

The basic data structures such as tags and tokens are the crucial part of morphological analysis process. For clarity and simpler use in other projects, these have been put in a separate library names Corpus2. Tags are assumed to be positional and represented in the colon-separated Morfeusz/Morfologik format with dot/underscore shorthands (see Section 2). Internally, tags are stored in a compact binary format. Corpus2 contains methods of manipulating these, as well as annotated token and sentence input and output (e.g. in the IPI PAN Corpus variant of the XCES format (Przepiórkowski, 2004)). Since tagsets that define valid tags can be non-trivial, a helper utility called `tagset-tool` has been created that allows inspecting a tagset and validating tags.

4.3 Maca

The Maca library proper contains two major modules: *analysis* and *conversion*. The analysis module uses morphological analysis sources such as plain text files, compiled SFST transducers (Schmid, 2005) or external libraries (currently only Morfeusz). An SFST transducer can be used either as a means of dictionary compression (we provide a convenience script to compile plain text dictionaries) or to introduce generalisation / guessing (in this case the user must write and compile a valid SFST program). Each analysis submodule interfaces with a particular morphology source and new submodules can easily be added to accomodate other libraries or transducer types, thanks to a simple plugin system. Interfacing alone is not enough, however, since there can be significant tagset differences or even token segmentation approaches in different morphological data sources. The conversion module enables some limited methods of conversion between tagsets, both on a simple per-tag level, as well as more complex rules operating on groups of annotated tokens, able to join or split tokens. In particular, with a slight tagset modification some of the segmentation ambiguities in Morfeusz can be folded into simple tag ambiguities.

Maca is configurable in a similar fashion to Toki, with simple text (INI-like) files. The used analysers can be grouped and applied in turn so that if one analyser fails to produce an analysis, another one will be used. This allows simple patching of known incorrect analyses in a large analyser by a small "patch" analyser. Furthermore, different groups of analysers can be used depending on the Toki-assigned token label. Support files such as tagsets or conversion rules are plain text files as well.

Two command-line utilities are available with Maca: `maca-analyse` and `maca-convert`. The former is used to process plain text, it uses Toki to tokenise the file and then analyse the tokens. `maca-convert` reads annotated tokens from one file and passes them through a tagset converter. In both cases the output is a sequence of tokens that can be written in any of the supported formats, e.g. XCES.

Internally Maca and its supporting libraries use ICU Unicode facilities for consistent treatment of Unicode data in the input texts. ICU also allows referring to convenient Unicode Properties, e.g. specifying `\p{Lu}` for upper-case letters of any script or `\p{Ps}` for opening punctuation, and efficient Unicode-aware regular expressions.

Maca does not directly depend on any non-free or component or any specific linguistic data. Morfeusz support is available as a plug-in that can be removed altogether, although it is used

in some default configurations. Thanks to the extendible architecture, more and more free resources can be integrated into the whole morpho-syntactic pipeline, which should eventually allow a completely free, high-quality analyser for Polish.

What is more, although targeted at Polish, these tools are actually quite universal. The only language-specific features are related to tag representation (the colon-separated positional format) and the choice of input/output formats (besides plain text, only XCES "`morph`" and "`pre_morph`" formats are currently supported).

5 Usage scenarios

In this section we describe some typical usage scenarios of our solution and provide some related details.

5.1 Compiling a simple analyser from existing data

This is the simplest scenario, where we have a morphological dictionary and we want to put this resource into a working processing pipeline. This can be achieved by the following steps:

1. using one of the provided Toki configurations or preparing one tailored to the required segmentation strategy,
2. compiling the morphological dictionaries into transducers with SFST (a convenience script is provided with Maca),
3. writing a simple Maca configuration that attaches fixed tags to punctuation and digits and the transducer to the rest of token types.

The configuration file may also be extended with some additional morphology sources, e.g. attaching multiple dictionaries that are consulted in the given order. What is more, some specific token types recognised by Toki may be attached completely different dictionaries. For instance, Polish acronyms may appear with inflectional endings after a hyphen (e.g. *SQL-a* stands for *SQL* in genitive case). It makes sense to treat such forms as whole tokens while splitting other hyphenated material. It is convenient to have Toki label such inflected acronyms with a specific token type and then use this information in Maca.

5.2 Using and patching Morfeusz

As noted above, Morfeusz is a non-free analyser with high-quality data for Polish. Nevertheless, it contains some erroneous entries, which cannot be corrected as the dictionary is closed. Maca allows to patch these errors by creating a list of correct forms that will supersede Morfeusz data.

What is more, Morfeusz is provided as a shared library with a rudimentary utility to pose queries. Maca compiled with Morfeusz plug-in allows to use the analyser as a convenient tool that produces XCES-compliant corpora from plain text or simple XML-based documents ("`pre_morph`").

As noted in Section 3, Morfeusz outputs DAG structures. Morfeusz plug-in is equipped with simple heuristics that when faced with segmentation ambiguity, chooses the shortest path (by default, warnings are issued when this happens).

5.3 Simple tag conversions

While a serious tagset conversion should arguably be performed off-line (e.g. conversion from Morfologik to the IPIC tagset), it is convenient to have simple tag conversions done on the fly. An example of such a usage scenario is related to the differences between the actual tagset of Morfeusz and the tagset of the IPI PAN corpus. The former contains some additional values of the gender attribute that are not present in the corpus; cf. the variants of the IPIC tagset as defined in Przepiórkowski (2004) and Przepiórkowski (2003). Maca allows to solve the problem by defining several tagsets and writing conversion routines (configuration files) that contain the necessary value mapping rules (such routines are included in the package).

Other simple usage of the on-the-fly tag conversion is reducing tags to contain grammatical class only (or some limited choice of attributes). This may be useful to test performance of some tagger or parser on reduced tags.

Some of the aforementioned segmentation ambiguities may be avoided by introducing a different segmentation strategy coupled with a modified tagset. If we choose to join the verb forms that in the IPIC tagset are split into several tokens (as discussed in Section 3), we are able to account for the whole ambiguity in the usual manner: by providing alternative interpretations of the

same token. The conversion module is able to perform limited joining and splitting of tokens if some required conditions hold. The affected tokens may be re-tagged by applying provided post-conditions. What is more, the Morfeusz plug-in allows for insertion of a conversion routine, which allows to process each graph path separately. If each of the converted paths consists of the same number of tokens and each token at respective position corresponds to the same string, the paths are *folded*. This results in linear sequence of tokens that may be output as such with no information loss. If the converted graph still contains segmentation ambiguity, the shortest path is selected.

We have prepared a configuration that allows to join the discussed verbs together and assign meaningful tags to them — an *intermediate tagset*. For instance, the form *miałem* whose Morfeusz analysis is depicted on Figure 1, in our tagset and configuration is analysed as one token with two interpretations: the noun interpretation (`subst` class) unchanged and the verb interpretation (`praet` and `aglt` tokens) joined together to form a finite for marked for preterite tense (`fin:sg:pri:imperf:prt:m1.m2.m3`). This is useful: if we run a morpho-syntactic tagger on input in the intermediate tagset, it will select one of these interpretations. The tagged file may be safely converted back to the original IPIC tagset as it contains no ambiguity now (the other interpretation should have been discarded).

6 Future work

The presented solution is already a practical system for processing Polish text. We plan to enhance the capabilities of the framework by introducing support for additional input/output formats. Another task is to provide Python wrappers for the basic functionality, making it easier to create fast prototypes of morpho-syntactic taggers while getting the full functionality of Maca.

However, most of the work will be devoted to gather more free morphological data by integrating the available resources and investigating tagset differences. The other planned projects assume the usage of Corpus2 and Maca APIs in the NLP software being built at our institute.

It would also be interesting to employ Maca for other languages.

References

Grefenstette, G. and Tapanainen, P. (1994). What is a word, what is a sentence? problems of tokenization. In *Proceedings of COMPLEX'94*, Budapest.

Hajnicz, E. and Kupść, A. (2001). Przegląd analizatorów morfologicznych dla języka polskiego. Technical Report 937, IPI PAN.

Miłkowski, M. (2010). Developing an open-source, rule-based proofreading tool. *Software: Practice and Experience*, 40:543–566.

Miłkowski, M. and Lipski, J. (2009). Using SRX standard for sentence segmentation in LanguageTool. In Vetulani, Z., editor, *Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 556–560, Poznań. Wydawnictwo Poznańskie, Fundacja Uniwersytetu im. A. Mickiewicza.

Obrębski, T. and Stolarski, M. (2006). *UAM Text Tools v0.90*.

Ogrodniczuk, M. (2003). Nowa edycja wzbogaconego korpusu słownika frekwencyjnego. In *Językoznawstwo w Polsce. Stan i perspektywy*, pages 181–190. Polska Akademia Nauk, Komitet Językoznawstwa, Uniwersytet Opolski, Instytut Filologii Polskiej, Opole.

Pooley, D. and Raya, R. M. (2008). Srx 2.0 specification.

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

Przepiórkowski, A. (2004). *The IPI PAN Corpus: Preliminary version*. Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Przepiórkowski, A. and Woliński, M. (2003). A flexemic tagset for Polish. In *Proceedings of Morphological Processing of Slavic Languages, EACL 2003*.

Schmid, H. (2005). A programming language for finite state transducers. In *Proceedings of the FSMNLP 2005*.

Vetulani, Z. (2000). Electronic language resources for Polish: POLEX, CEGLEX and GRAMLEX. In Gavrilidou, M., Carayannis,

G., Markantonatou, S., Piperidis, S., and Stainhaouer, G., editors, *LREC2000 Proceedings*, pages 367–374, Paris. ELRA.

Woliński, M. (2006). Morfeusz — a practical tool for the morphological analysis of Polish. In Kłopotek, M. A., Wierzchoń, S. T., and Trojanowski, K., editors, *Proceedings of IIPWM'06*, pages 511–520, Ustroń, Poland. Springer-Verlag, Berlin.