Gobio and PSI-Toolkit: Adapting a deep parser to an NLP toolkit

Paweł Skórzewski

Adam Mickiewicz University
87 Umultowska Street, Poznań, Poland
pawel.skorzewski@amu.edu.pl

Abstract

The paper shows an example how an existing stand-alone linguistic tool may be adapted to a NLP toolkit that operates on different data structures and a different POS tag-set. PSI-Toolkit is an open-source set of natural language processing tools. One of its main features is the possibility of incorporating various independent processors. Gobio is a deep natural language parser that is used in the Translatica machine translation system. The paper describes the process of adapting Gobio to PSI-Toolkit, namely the conversion of Gobio's data structures into the PSI-Toolkit lattice and opening Gobio's rule files for edition by a PSI-Toolkit user. The paper also covers the technical issues of substituting Gobio's tokenizer and lemmatizer by processors used in PSI-Toolkit, e.g. Morfologik.

Keywords: natural language processing, parsing, data structures

1. Introduction

1.1. PSI-Toolkit

PSI-Toolkit is an open-source natural language processing framework (Graliński et al., 2012; Jassem, 2012). It has been developed on Adam Mickiewicz University in Poznań, by the Information Systems Laboratory team (*Pracownia Systemów Informacyjnych* in Polish, hence the toolkit's name). It consists of a set of tools (called processors) that can be run as a tool-chain. There are three kinds of processors in PSI-Toolkit: readers, writers and annotators. Firstly, a reader reads data from an external source, e.g. an XML file, and inserts the data into the main data structure, called PSI-lattice. Then, a chain of annotators is applied. Among annotators, there are a variety of natural language processing tools, such as segmenters, tokenizers, lemmatizers, parsers, machine translators and spellcheckers. Finally, a writer writes the result in the desired format to the screen or to the file.

PSI-Toolkit has a modular structure that allows adding new processors. A user can apply the tool (e.g. lemmatizer) of their choice. All annotators used in PSI-Toolkit share the same data structure: the PSI-lattice, which contains all the information about the current state of processed data.

PSI-Toolkit can be operated in two ways: through the command line interface or through a web service. It is also distributed in packages for main Linux distributions (Ubuntu, Debian, Linux Mint, Arch Linux).

1.2. Gobio

Gobio is a natural language parser. It is a deep parser, i.e. its result is the full syntactic structure of a given sentence, indicating the syntactic roles of constituents (in contrast to shallow parsers). The parsing result has the form of the parse tree that shows dependencies between constituents. It is a type of a chart parser that uses a variant of Cocke-Younger-Kasami algorithm. Gobio uses a tree-generating binary grammar with weights (Graliński, 2006). Gobio was originally developed as a deep parser for German language in the Translatica machine translation system (Jassem, 2006) and it can be used for parsing various natural languages. Sets of rules for different languages, including German, Polish, Russian and English, have already been designed for the parser. Rules and weights for some languages (Polish, Russian) have been created manually, whereas rules and weights for other languages (German, English) have been automatically extracted from corpora.

As the parser was intended for a specific machine translation system, Gobio is heavily integrated with (and dependent on) other mechanisms and formats belonging to Translatica, such as its segmenter, its lemmatizer or its tagsets.

1.3. Goals

At the time of writing this paper, besides Gobio, PSI-Toolkit includes two other parsers: Puddle and Link Grammar Parser. Puddle is a shallow parser based on the Spejd parser originally developed at IPI PAN (Przepiórkowski and Buczyński, 2007). Link Grammar Parser is a parser developed on Carnegie Mellon University (Sleator and Temperley, 1995) that uses a link grammar, where the result of parsing has a form of links between words of the given sentence. However, the grammar has the option to produce output in a form of a simple parse tree. Both parsers are multilingual: they can parse different languages if parsing rules are provided. However, no Link Grammar rules have been developed for the Polish language yet. Neither of these parsers can produce a detailed parse tree.

We wanted to include the Gobio parser in PSI-Toolkit in order to offer a deep parser in the toolkit. Furthermore, we planned to include also the machine translation module from Translatica in PSI-Toolkit. For this reason, the inclusion of Gobio in PSI-Toolkit would be not only complementary, but even necessary.

PSI-Toolkit is a flexible modular environment that allows including different natural language processing tools.

2. Main issues

In order to make the operation of including Gobio parser into PSI-Toolkit successful, we had to solve a few problems. Both systems have been written mainly in C++, but there are many differences and incompatibilities between them. PSI-Toolkit and Translatica use different data structures and different tagsets. Translatica is a system specifically geared towards machine translation, and thus preparatory activities like sentence splitting, tokenization or morphological analysis are executed by tools heavily integrated with the system. We had to separate those tools from the parser itself in order to adapt Gobio to the modular system like PSI-Toolkit.

2.1. PSI-lattice and Gobio's chart

PSI-lattice is the main data structure used in PSI-Toolkit. It is a kind of a word lattice used in natural language processing (Dyer et al., 2008). It consists of vertices, marking positions in the input text, and edges that link them and span appropriate substrings of text. During text processing the reader creates the lattice, then the subsequent annotators adds their edges to it, finally the writer prints out the formatted lattice content. The basic unit PSI-lattice is a single character, i.e. lattice's vertices are the inter-character points.

The Gobio parser has been build atop of a different data structure: the chart. The chart is a structure similar to lattice, basic units are words (rather than characters): the chart vertices are the word boundaries.

2.2. Storing edge information

Each edge in the PSI-lattice consists of the following elements:

- Source and target vertices. They indicate the beginning and the end of text that is spanned by the edge.
- Annotation results attached to the edges. They are stored as strings indicating edge category, edge-spanned text and processor-specific annotations, e.g. morphosyntactic features, such as case, gender, number, person, tense etc.
- Layer tags attached to the edges. They express some meta-information, e.g. edge type, name of the processor or tagset used.
- Partitions. They indicate which edges were used to build the given edge.
- Scores (weights). These are floating-point values assigned to the edges. A score can refer to the whole edge, but there is also a possibility to assign different scores to different partitions. Scores can be used as indicators of edge likelihood in the process of building a parse tree.

Edges of the original Gobio's chart have also source and target vertices (but they correspond to word boundaries, not inter-character points), and scores with floating-point values. There is no such thing like layer tags in the chart. Each edge also stores information about its partitions. The main difference is the structure of the item responsible for storing information about edge category and attributes. The chart stores this information in elements called the attribute-value matrices (AV-matrices). Categories and attribute values are stored in AV-matrices as integers; the actual values corresponding to these numbers are stored in structures called registrars.

2.3. Tagsets

At present, PSI-Toolkit uses two morphological analyzers. One of them is Morfologik, a morphological analyzer developed by Miłkowski (2010). Morfologik uses a tagset based on that of the National Corpus of Polish (Przepiórkowski, 2009). The second one is Lamerlemma, a simple lemmatizer developed specially for PSI-Toolkit. Lamerlemma uses the same tagset as Morfologik.

Gobio uses its own tagset. It differs significantly from the Morfologik tagset. It takes into account the lexicon information such as verb valencies, which is not provided in either Morfologik or Lamerlemma.

2.4. Sentence splitting, tokenization and morphological analysis

PSI-Toolkit has a modular structure. Sentence splitters, tokenizers, morphological analyzers and other tools are independent processors. This approach is flexible because it allows the user to choose the modules that are most suitable for a given task. The approach used in Translatica is different. It is a monolithic system, focused on machine translation. Translatica segmenter, tokenizer and lemmatizer are integral parts of the system.

We had to separate those processors from the actual parser and modify Gobio in a way that allows using other tools than integrated in Translatica.

2.5. Parsing results

In the original Gobio parser, the parsing result was created as follows. First, the given sentence was inserted into the chart. Then, a module called *combinator* tried to combine edges form the chart to build new edges. In this way, the forest that contained all possible partial parses was created. Finally, a module called *chooser* selected the edges for the final parse tree, according to tree-generating binary grammar rules. The final parse tree was not inserted into the chart, but rather it was stored in a different, special data structure.

PSI-toolkit was created with the idea that both intermediate results and the final result should be stored explicitly in the PSI-lattice. This meant that we needed a mechanism that would insert the Gobio's parsing result into the PSI-lattice.

3. Solutions

3.1. Zvalues

Zvalue is a special data type that can store various kinds of values: numbers, strings, vectors, hashes or even parse trees. Zvalue is a very efficient data type, because it is implemented simply as a specially wrapped void pointer.

All values of annotation item attributes in PSI-Toolkit are implemented as zvalues, which allows storing both numbers and strings, or even more sophisticated values. We had to modify Gobio's structures to operate on zvalues instead of integer values where necessary.

3.2. PSI-lattice wrapper and AV-AI converter

Translatica/Gobio's chart and PSI-lattice have different interfaces. We wrote a wrapper that hid PSI-lattice's interface so PSI-lattice could be handled by Gobio as if it

was a Translatica chart. This had to include conversion between AV-matrices and annotation items.

We wrote a special class to convert AV-matrices into annotation items and its category and attribute values: AV-AI converter.

The drawback of this solution is that it causes a slight slowdown of the parser performance, but it was necessary to integrate the Gobio parser with the PSI-lattice datastructure.

3.3. Tagset converter

In order to deal with different tagsets, we needed a converter between Morfologik tags and Gobio/Translatica tags, but we proposed a more general solution. We built a tagset converter that can convert between any two tagsets provided that conversion rules are specified. Tagset converter is an independent tool in PSI-Toolkit, and it can be linked with different tools of the framework. Thus the proposed solution facilitates future system expansion and work with other tagsets.

In addition to the conversion rules, a tagset converter's rule file contains the specification of the layer tags indicating which kinds of edges should be converted. There are two types of rules: simple substitutions and if-then clauses. Substitution rules allow replacing edge categories, attribute names and values. If-then clauses allow conditional substitutions. A syntax of these clauses allows specifying complex conditions on edge categories, values of the attributes or even edge texts.

A sample tagset converter's rule file is shown in Fig. 1.

```
# Morfologik to Translatica
# tagset conversion rules.
@source morfologik-tagset
@target gobio-tagset
@tags   lexeme  form
@cat    adj przym
@attr   number  L
@val    sg  1
@val    pl  2
CAT=rzecz, niż >> CAT=rzecz|przyim
CAT=rzecz, L=1, Rp=mo >> R=mż, R1=$Rp
```

Fig. 1: A sample tagset converter's rule file

Lines beginning with # are treated as comments. The two lines beginning with `@source` and `@target` indicate the source tagset and the target tagset respectively (or in fact their layer tags). The line beginning with `@tags` indicates layer tags that should be preserved in copied edges. Other lines beginning with `@` contain simple substitution rules. Keywords `@cat`, `@attr` and `@val` indicate which annotation type (category, attribute or value) the substitution rule concerns.

Nonempty lines that begin with symbols other than # or `@` contain if-then clauses. Each clause consists of conditions and commands. The conditions block is separated from the commands block by >>.

The conditions are separated by commas. If a condition has a form of a single word, it is satisfied when the edge text is equal to this word. Otherwise, a condition has a form of an equation. It is satisfied when the value of the attribute to the left is equal to the value to the right, or if there is a keyword `CAT` to the left and edge category equals the value to the right. For the commands to be executed, all conditions have to be satisfied.

The commands are also separated by commas. All commands have the form of equations. If the equation does not contain the `$` symbol, a value of the attribute to the left is substituted by the value to the right in the newly formed edge. If there is a keyword `CAT` to the left, the edge category is changed to the value to the right. If a string to the right is preceded by the `$` sign, it means that the value of the attribute to the left should be substituted by the value of the attribute preceded by `$`. If there are several values separated by a vertical line (`|`) to the right, it means that the edge should be cloned and each copy should get one of the given values.

This system, yet simple, allows writing even complex conversion rules. Thanks to the tagset converter, we can convert successfully every annotation expressed in the Morfologik tagset to the Translatica/Gobio tagset.

3.4. Mapper and joiner

Because PSI-Toolkit lemmatizers provide no lexicon information, we needed a simple lexicon and a valency dictionary to feed Gobio. So we created *mapper* – a universal lexicon for generic mapping tasks. Given a valency dictionary, mapper generates the valencies. Mapper is an independent tool in the PSI-Toolkit framework.

Another important tool that had to be created in order to provide lexicon information for the parser was *joiner*. Joiner is a simple tool that produces a Cartesian product of the specified two sets of edges. In this case, it allows creating edges that are combinations of forms generated by a lemmatizer and valencies generated by the mapper. Such edges constitute the input for the Gobio parser.

3.5. Putting the final parse tree into the lattice

We provided a simple code that adds new edges corresponding to edges of the final parse tree to the lattice after the parsing is done. We also had to make several changes to the existing chooser.

3.6. Machine translation

At the moment, there are two tree-to-string decoders in PSI-Toolkit: Transferer and Bonsai. They can be applied after parsing to create translations (a translator is a pipe consisting of a sentence splitter, a tokenizer, a lemmatizer, a parser and a tree-to-string decoder).

Transferer is a rule-based machine translation system. It has been created to be compatible with Gobio. It uses rules expressed in a special programming language dedicated to the manipulation on parse trees.

Bonsai is a tree-to-string decoder dedicated to the statistical machine translation. It operates on parsed input so it requires a parser to be applied before.

Both Transferer and Bonsai combined with Gobio create a decent machine translation that can be used for various language pairs (e.g. Polish-English, Polish-Spanish), depending on the available parsing and transfer rules.

4. Conclusions and future work

Adaptation of Gobio to PSI-Toolkit was a challenging project that has brought us a lot of experience. We had to face many difficulties arising during adaptation. The resulting solutions, such as zvalues, PSI-lattice wrapper or AV-AI converter, can be the basis for components of further projects. Integration of Gobio with PSI-Toolkit was also an impulse for the creation of a number of useful tools like mapper, joiner and tagset converter.

In PSI-Toolkit, Gobio can be connected with one of the translators and create a machine translation system.

We claim that a lattice data structure assumed for PSI-Toolkit, has proved helpful for the adaptation of an external NLP tool.

The case described here confirms the postulate for architecture modularity in NLP systems. The modularity of PSI-Toolkit makes it possible to adapt various external resources (including Gobio), whereas the lack of this feature in the Translatica architecture has brought about several problems that had to be solved in order to extract the parser from the MT system.

We plan to further develop the Gobio parser within the PSI-toolkit framework. This will include adding parsing rules for more languages. We also plan to implement alternative parsing algorithms.

Acknowledgements

This paper is based on research funded by the Polish National Science Centre (decision no. DEC-2011/01/N/ST6/02032).

References

Dyer, C., Muresan, S. and Resnik, P. (2008). Generalizing word lattice translation. In: McKeown, K., Moore, J.D., Teufel, S., Allan, J. and Furui, S. (Eds.): *ACL, The Association for Computer Linguistics*, pp. 1012–1020.

Graliński, F. (2006). Some methods of describing discontinuity in Polish and their cost-effectiveness. *Lecture Notes in Artificial Intelligence* 4188, pp. 69-77. Heidelberg: Springer.

Graliński, F., Jassem, K. and Junczys-Dowmunt, M. (2012). PSI-Toolkit: Natural language processing pipeline. *Computational Linguistics – Applications*, Heidelberg: Springer.

Jassem, K. (2006). *Przetwarzanie tekstów polskich w systemie tłumaczenia automatycznego POLENG*. Poznań: Wydawnictwo Naukowe UAM.

Jassem, K. (2012). PSI-Toolkit – how to turn a linguist into a computational linguist. *TSD 2012. Lecture Notes in Computer Science*, Springer.

Miłkowski, M. (2010). Developing an open-source, rule-based proofreading tool. *Software: Practice and Experience* 40(7), pp. 543-566.

Przepiórkowski, A. (2009). A comparison of two morphosyntactic tagsets of Polish. In: Koseska-Toszewa, V., Dimitrova, L. and Roszko, R. (Eds.): *Representing Semantics in Digital Lexicography: Proceedings of MONDILEX Fourth Open Workshop*, Warsaw, pp. 138-144.

Przepiórkowski, A. and Buczyński, A. (2007). SPADE: Shallow Parsing and Disambiguation Engine, *Proceedings of the 3rd Language and Technology Conference*, Poznań.

Sleator, D.D. and Temperley, D. (1995). Parsing English with a link grammar. *arXiv preprint cmp-lg/9508004*.