PSI-Toolkit — an Extensible and Tightly Integrated Set of NLP Tools

Krzysztof Jassem, Filip Graliński, Marcin Junczys-Dowmunt, Paweł Skórzewski, Roman Grundkiewicz, Marcin Walas, Rafał Jaworski, Tomasz Dwojak

Adam Mickiewicz University
ul. Wieniawskiego 1, 61-712 Poznań, Poland
{jassem,filipg,junczys,pms,romang,walas,rjawor,t.dwojak}@amu.edu.pl

Abstract

The paper reports on PSI-Toolkit, an extensible set of NLP tools developed at Adam Mickiewicz University in Poznań. All processors of the toolkit operate on a common data structure called PSI-lattice. This feature allows the seamless incorporation of NLP tools created by other researchers. PSI-Toolkit is licensed under LGPL, which allows for unrestricted commercial use.

1. Introduction

PSI-Toolkit (Graliński et al., 2012) is a set of linguistic tools developed within a project financed by the Polish Ministry of Science and Higher Education between 2011 and 2013. The toolkit consists of a set of tools that process natural language texts in Polish, English, German and a few other languages[1]. It focuses on machine translation (both rule-based and statistical), but also includes tools required in other natural language processing tasks, such as sentence splitting, tokenization, lexical and morhological analysis or syntactic parsing.

PSI-Toolkit attempts to ensure extensibility by unifying the communication between NLP modules. This is executed by the design of a universal data structure that can be accessed by any type of an NLP tool.

This paper is organised as follows: Section 2. describes the toolkit, focusing on the data structure. Section 3. presents the target audience. Section 4. compares PSI-Toolkit with other NLP toolkits. Summary and ideas for future work are listed in Section 5..

2. System Description

PSI-Toolkit consists of a set of NLP tools called processors: readers, annotators, and writers. In a typical use-case processors are combined into a pipeline, which starts with a reader, followed by a sequence of annotators, and ends with a writer. All processors in a pipe are separated by the exclamation marks (chosen due to its similarity to the pipe symbol):

```
a reader ! a sequence of annotators
  ! a writer
```

A reader initializes the main data structure, the so-called PSI-lattice, from an external source of information. An annotator (e.g. a tokenizer, a lemmatizer, a parser) adds new annotations to the PSI-lattice, usually based on information added by preceding annotations. Finally, a writer produce an output in desired format.

2.1. PSI-lattice

All PSI-Toolkit processors operate on a common lattice-based data structure, called *PSI-lattice*. As described by Graliński et. al (Graliński et al., 2012) PSI-lattice is defined as a graph where vertices represent the input string (one vertex per character) and edges represent annotations.

A PSI-lattice provides a one-size-fits-all data structure that contains all annotations generated by all processing tools in a pipeline. The structure provides a standardized way to pass around alternative interpretations. This allows for *delayed disambiguation*, i.e. a higher-order tool may disambiguate alternatives provided by a lower-order tool (Graliński et al., 2012). A PSI-lattice can be constructed for entire corpora as a single data structure or created and processed in a line-by-line fashion.

2.2. PSI-Toolkit Processors

The current list of PSI-Toolkit processors (39 in total) is available in documentation[2]. The function of each PSI-Toolkit processor can be customized by means of switches (options), again mimicking command line pipelines.

Readers read data from the input stream, retrieve textual information and initiate PSI-Lattice. At present, PSI-Toolkit readers support raw text (`txt-reader`), Portable Document Format (`pdf-reader`), various XML-based formats (e.g. XHTML, OpenDocument, Open XML formats — `apertium-reader`), the UTT format (Obrębski and Stolarski, 2006) (`utt-reader`) and the NKJP format (Przepiórkowski et al., 2008) (`nkjp-reader`).

A special processor — `guessing-reader` — guesses the input file format and calls an appropriate reader. It is used by default when a processing pipeline does not include a reader.

PSI-Toolkit **annotators** add new edges to PSI-lattice spanning over characters. They are categorised into several groups:

- Tokenizers: a sentence splitter using SRX (Segmentation Rules eXchange[3]) standard called

[1] Different tools usually support different languages, for details refer to the official documentation at `http://psi-toolkit.amu.edu.pl/help/documentation.html`.

[2] `http://psi-toolkit.amu.edu.pl/help/documentation.html`

[3] `http://www.gala-global.org/oscarStandards/srx/srx20.html`

`srx-segmenter` and multi-language tokenizer (`tp-tokenizer`).

- Lemmatizers and lexicons: a dictionary-based multi-language lemmatizer called `lammerlema`, a bilingual lexicon for machine translation (`bilexicon`) and a universal lexicon for generic mapping tasks (`mapper`).
- Taggers: `metagger`, a part-of-speech tagger based on the maximum entropy classification; `inflector`, a tool for generating inflected forms of lemmatized words and `lang-guesser`, an automatic language identification tool.
- Parsers: the Puddle rule-based shallow parser (`puddle`) modeled on Spejd parser (Buczyński and Przepiórkowski, 2008), the syntax parser (Skórzewski, 2013) for Polish (`gobio`) incorporated from *Translatica* (a commercial machine translation system), and the Link Grammar Parser (Sleator and Temperley, 1993) (`link-parser`).
- Translators: a rule-based machine translation system, again adapted from *Translatica* (`translator`), a syntax-based statistical machine translation application Bonsai (`bonsai`).

Writers output all or selected PSI-lattice edges in a required format. Writers can convert a PSI-lattice into a human-readable way (`simple-writer`), produce tagged output (`bracketing-writer`), draw simple graphs (`gv-writer`), generate the JSON format (`json-simple-writer`), or print the complete information stored in PSI-lattice.

If no writer is specified, the `simple-writer` selects the most appropriate edges to be printed out based on the last annotator in the sequence. For example, if a pipeline ends with a tokenizer, then the writer returns only tokens separated by whitespaces.

2.3. Use Cases

Language tools of PSI-Toolkit can be accessed in various ways: the *psi-pipe* command-line tool, bindings for Perl and Python programming languages, a graphical user interface[4] and JSON API[5].

3. Target Audience

PSI-Toolkit is addressed to three types of users: NLP professionals, linguists and NLP or IT students (Jassem, 2012). Experienced Unix users apply the command-line tool *psi-pipe* in combination with other Unix commands. Such users are prompted to compile the source code and adjust existing processors to their requirements (e.g. by adding new rule files or by training their own statistical models).

For less computer-savvy users, such as linguists or translators we developed a web interface which offers the same functionalities as the command-line tool in a more user-friendly way. The web portal includes tutorial and end-to-end usage examples.

PSI-Toolkit command-line tools is also intended to serve educational purposes. We believe that the auto-completing function (see Section 4.2.) will help students understand the general issues of natural language processing.

4. Comparison to Other NLP Systems

We shall limit the comparison to the following well-known toolkits: GATE (Cunningham et al., 2011), NLTK (Loper and Bird, 2002), Apertium (Forcada et al., 2011) and Stanford NLP (Manning and Schütze, 1999). The comparison will focus on the aspects of extensibility, usability and possibility of commercial use.

4.1. Flexibility and Extensibility

To the best of our knowledge PSI-Toolkit is distinguished from other toolkits by its universal, internal data structure, which organizes communication between its components. Other toolkits define input and output textual data formats for each tool instead.

PSI-lattice allows for easy extension. Adapting a new tool to PSI-Toolkit requires no more than designing a mechanism of reading and writing data from and into the data structure (Jassem, 2013). We have succesfully tested this procedure with the Link Grammar (Sleator and Temperley, 1993). PSI-Toolkit allows users to provide custom linguistic resources for various tools. The same feature is applied in Apertium — limited for Machine Translation.

Another PSI-Toolkit feature, boosting its flexibility, is the *psi-pipe* console interface. It enables the use of PSI-Toolkit processors together with — ofter underestimated — Unix text processing commands.

4.2. Usability

Like other NLP toolkits, PSI-Toolkit provides detailed documentation with tutorials and code snippet generators. Unlike NLTK, PSI-Toolkit does not require its users to perform any computer programming.

Using PSI-Toolkit pipelines is facilitated by a novel auto-completing function. The function guesses how a processor pipeline should be completed to make it a working command. The completion concerns completion of names of processors or required switches. The following examples should clarify the idea:

- Suppose a user wants to parse a text in an unspecified language and format. The command `parse` suffices to execute the task. The **processor auto-completing feature** expands the user's command to the following pipeline:

```
guess-format ! guess-language
  ! segment ! tokenize ! lemmatize
  ! parse
```

- The **switches are auto-completed** in order to fit to the first user choice, e.g. the first pipeline is auto-completed to the second one:

[4] `http://psi-toolkit.amu.edu.pl/`

[5] `http://psi-toolkit.amu.edu.pl/json.psis`

```
tokenize --lang pl ! segment
tokenize --lang pl ! segment
  --lang pl
```

- Another functionality intended for the facilitation of use is **aliases**[6], i.e. alternative names for processor sequences (including their options). For instance, `translate-plen` is an alias for:

```
gobio --lang pl ! bilexicon
  --lang pl --trg-lang en
  ! transferer --lang pl
  --trg-lang en
```

4.3. Commercial Use

PSI-Toolkit is licensed under LGPL, which allows for full commercial use. This is rarely seen in other toolkits (Apertium and Stanford NLP are licensed under GPL). So far, two commercial projects based on PSI-Toolkit have already been completed.

5. Summary

The paper describes an open-source NLP toolkit, whose unique feature is a universal data structure. The goal of the system developers is to gather, in one environment, NLP tools designed in various research centers. This should result in new cases of co-operation cases between NLP researchers. The toolkit simplifies a potential use of various types of NLP tools in one complex system. Future work will focus on integration of new tools as well as efficiency and scalability issues.

6. References

Buczyński, Aleksander and Adam Przepiórkowski, 2008. Spejd demo: An open source tool for partial parsing and morphosyntactic disambiguation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA).

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters, 2011. *Text Processing with GATE (Version 6)*.

Forcada, Mikel L., Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez Felipe Sánchez-Martínez, and Francis M. Tyers, 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*:127–144. Special Issue: Free/Open-Source Machine Translation.

Graliński, Filip, Krzysztof Jassem, and Marcin Junczys-Dowmunt, 2012. PSI-Toolkit: Natural language processing pipeline. *Computational Linguistics - Applications*, 458:27–39.

Jassem, Krzysztof, 2012. PSI-Toolkit — how to turn a linguist into a computational linguist. In Petr Sojka, Ales Horák, Ivan Kopecek, and Karel Pala (eds.), *Proceedings of 15th International Conference on Text, Speech and Dialogue*, Lecture Notes in Computer Science.

Jassem, Krzysztof, 2013. PSI-Toolkit — an open architecture set of nlp tools. In Zygmunt Vetulani and Hans Uszkoreit (eds.), *Proceedings of the 6th Language and Technology Conference*. Poznań.

Loper, Edward and Steven Bird, 2002. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02. Association for Computational Linguistics.

Manning, Christopher D. and Hinrich Schütze, 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press.

Obrębski, Tomasz and Michał Stolarski, 2006. UAM text tools — a flexible NLP architecture. Proceedings of LREC 2006.

Przepiórkowski, Adam, Rafał L. Górski, Barbara Lewandowska-Tomaszczyk, and Marek Łaziński, 2008. Towards the National Corpus of Polish. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC 2008)*.

Skórzewski, Paweł, 2013. Gobio and PSI-Toolkit: Adapting a deep parser to an NLP toolkit. In Zygmunt Vetulani and Hans Uszkoreit (eds.), *Proceedings of the 6th Language and Technology Conference*. Poznań.

Sleator, Daniel D. and Davy Temperley, 1993. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*.

[6] `http://psi-toolkit.amu.edu.pl/help/aliases.html`