PSI-Toolkit - a customizable set of linguistic tools

Krzysztof Jassem

Adam Mickiewicz University
Poznań, Poland
jassem@amu.edu.pl

Abstract

The paper describes PSI-Toolkit, a set of NLP tools designed within a grant of Polish Ministry of Science and Higher Education. The main feature of the toolkit is its open architecture, which allows for the integration of NLP tools designed in independent research centres. The toolkit is widely customizable: PSI-Toolkit processes a variety of text formats and languages. There are no constraints on annotation tag-sets. PSI-Toolkit annotation pipelines may consist of both PSI-Toolkit annotators and external tools.

1. Introduction

PSI-Toolkit is a set of NLP tools designed within a grant of Polish Ministry of Science and Higher Education. The primary goal of the project is to ensure public and free access to the set of NLP tools designed in the Laboratory of Information Systems (PSI is the Polish abbreviation for the name of the laboratory) at Adam Mickiewicz University in Poznań, Poland. The architecture of the toolkit is designed in such a way as to enable incorporation of NLP tools developed at other NLP centers. Users can combine PSI-tools with external applications in one processing pipeline. This is facilitated by tag-converters, which allow for the data exchange between annotators operating on different tag-sets.

PSI-Toolkit may be personalized according to users' needs. Users may customize PSI-Toolkit in two ways: by selecting run options of PSI-Toolkit annotators or by substituting annotation rules.

The data structure used in PSI-Toolkit is that of a lattice, where the edges span over the characters of the processed texts. Each annotator of the processing pipeline adds new edges to the existing structure (see (Graliński et al., 2012) for more details).

The functionality of PSI-Toolkit attempts to combine selected features of well-known NLP toolkits. We follow the Stanford Natural Language Processing Group (`nlp.stanford.edu`) in letting a user run PSI-Toolkit from a command line. Just like in the NLTK toolkit (Bird et al., 2009) and UIMA (`uima.apache.org`) we want programmers to be capable of building programs that call PSI-Toolkit annotators. We would like users to apply pipelines, as e.g. in GATE (`http://gate.ac.uk/`). Finally, we aim at encouraging pure linguists to use PSI-Toolkit, by delivering a friendly web-service - this is motivated by Apertium (`www.apertium.org`).

The full range of PSI-Toolkit functions is offered in Linux packages (prepared for Ubuntu, ArcLinux, Debian or Mint distributions). Most functions are also available in the web-service: `psi-toolkit.amu.edu.pl`. Fig. 1 shows the main window of the service.

A standard PSI-Toolkit command is formed as a pipeline of annotators. An example of a pipeline is shown in Fig. 2.

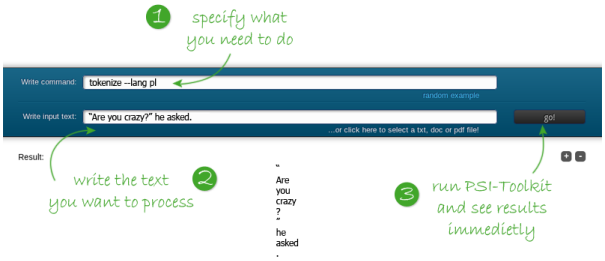

Figure 1: PSI-Toolkit web-service

read | segment | tokenize | lemmatize | write

Figure 2: Example of a PSI-Toolkit pipeline

2. Customizing PSI-Toolkit annotators

This section reports on PSI-Toolkit annotators and the ways of customizing them.

2.1. Readers

Readers are PSI-Toolkit annotators that extract texts from input files, split them into characters and build initial edges of the PSI-lattice that span over each character and the whole textual fragments. A user may customize the process in two ways: by running one of several types of readers delivered by PSI-Toolkit or by selecting desired options of the chosen reader. The following readers are delivered by PSI-Toolkit:

txt-reader - reads plain text from a file or standard input

apertium-reader - reads text from various markup formats

pdf-reader - reads PDF files

nkjp-reader - reads texts from the Polish National Corpus XML files (Przepiórkowski, 2011)

utt-reader - reads files in the UTT (UAM Text Tools) format (`http://http://utt.amu.edu.pl/files/utt.html`)

psi-reader - reads files in the PSI-Toolkit format (`http://psi-toolkit.wmi.amu.edu.pl/help/psi-format.html`)

Command
lang-guesser | simple-writer –tag !en
Input
Die Familie Grimm war in Hanau beheimatet.
Jacob Ludwig Carl Grimm, born on 4 January 1785, was 13 months older than his brother Wilhelm Carl Grimm.
Obaj bracia byli członkami Akademii Nauk w Berlinie i uczonymi (językoznawcami), o znacznym dorobku.
Output
Jacob Ludwig Carl Grimm, born on 4 January 1785, was 13 months older than his brother Wilhelm Carl Grimm.

Figure 3: Extraction of a fragment written in a specified language

A user may decide not to define the reader, leaving the task to the PSI-Toolkit. A special processor called *guessing-reader* guesses the input format.

Readers may be run with various options depending on the type of the reader: *text-reader* processes text as the whole (the *whole-text* option), which is recommended for short texts, or *line-by-line*, recommended for long texts. The options of *apertium-reader* allow for the processing of *rtf*, *html*, *Open Office* or *Microsoft Office* files, with the special option (*unzip-data*) intended for compressed file formats, such as *pptx* or *xslx*.

An interesting feature of PSI-Toolkit readers is the ability to read annotations of external tools. Currently PSI-Toolkit supports two annotation formats for Polish texts besides the PSI-toolkit format: *nkjp-reader* processes XML files that contain annotations of sentences in the *nkjp* corpus (Przepiórkowski et al., 2012) and *utt-reader* reads a specialized format that results from annotating text with *utt* tools (Obrębski, 2005). The readers convert external annotations into the edges of the PSI-Toolkit lattice. This feature enables co-operation of tools designed independently, e.g. a text tokenized within the *nkjp* corpus may be parsed syntactically by a PSI-Toolkit parser; a sentence parsed by an *utt* dependency parser may be displayed by means of the PSI-Toolkit graphical writer; a corpus parsed by the *utt* parser may serve for the training of the syntax-based PSI-Toolkit statistical translator.

2.2. Processing the Language of the Text

PSI-Toolkit puts no constraints on the language of the text - as long as it is UTF8-coded. If no language is specified, a special processor, called *lang-guesser*, tries to recognize the language based on bigram models. *lang-guesser* may be used for the extraction of foreign fragments from a text, as shown in Fig. 2.2. There, *lang-guesser* creates an edge tagged with the language code *!en*, spanning over the English fragment of the text. The option *–tag !en* of *simple-writer* limits the display of the text to fragments labeled with the *!en* tag.

Users may customize the PSI-Toolkit annotators to process the text according to rules specific for a language. For example, setting the *–lang* option to *-en* makes the *segmenter* use the sentence-splitting rules specific for the English language.

Command
tokenize | aspell –lang en
Input
I enjoy travleling
Output
I
enjoy
travleling|travelling|traveling|travailing|travellings|ravelling

Figure 4: A use-case for a spell-checker

2.3. Spell-Checking

Spell-checking is customized by choosing the language of the input text. PSI-Toolkit allows for using external lexicons for spell-checking. Currently, PSI-Toolkit applies the *aspell* lexicon that supports over 80 languages. Each *aspell* suggestion for an unrecognized token is converted into a PSI-lattice edge spanning over the token. A user of the local version of the toolkit may personalize spell-checking by adding new lexicons either for supported or unsupported languages. Fig. 2.3. shows a PSI-Toolkit use-case for a spell-checker.

2.4. Tokenization

tokenizer splits texts into tokens according to rules defined in an SRX (Segmentation Rules Exchange) file. PSI-Toolkit supports segmentation for 9 languages (and one default language). Tokenization may be customized by choosing the language and/or the maximum length of the token. A user of the local version may deliver a personalized SRX file (this is done via the *rules* option of *tokenizer*) either for supported or unsupported languages.

2.5. Sentence-Splitting

segmenter splits texts into segments (i.e. sentences) according to rules defined in an SRX file. PSI-Toolkit supports segmentation for 9 languages (and one default language). Sentence-splitting may be customized by choosing the language and/or the maximum length of the sentence.
A user of the local version may deliver a personalized SRX file (this is done via the *rules* option of *segmenter*) either for supported or unsupported languages.

2.6. Lemmatization

PSI-Toolkit supports lemmatizers for 6 languages: Polish, English, German, Italian, French and Spanish. A user of the local version may create and use a personalized lemmatizer. It suffies to deliver the lemmatization rules in the form of 3 files:

path to the lexicon (in the binary or plain text format)
path to the text file containing part-of-speech information
path to the text file containing morphological information

2.7. Tag-Set Conversion

PSI-Toolkit puts no constraints on the format of the information returned by the lemmatizer. It is allowed for different PSI-tools called in the same pipeline to operate on different tag-sets. For example, the Polish lemmatizer supported by PSI-Toolkit is based on the *morfologik* (`http://sourceforge.net/projects/morfologik/files/`) tag-set, whereas the deep parser operates on a different tagset developed for Tree-generating

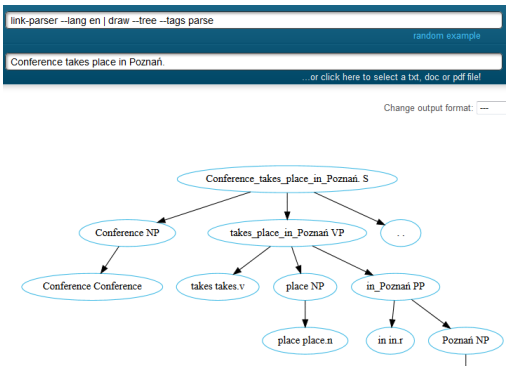

Figure 5: Link Grammar adapted in PSI-Toolkit

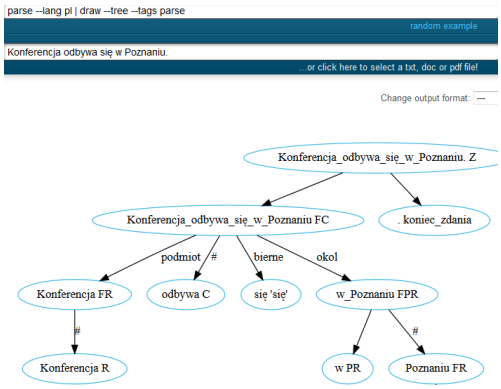

Figure 6: Deep Parser for Polish

Binary Grammar (Graliński, 2007). Still, the two tools may be called in one pipeline using a tag-set converter that maps the tag-sets. The following pipeline draws a resulting syntactic tree for an input sentence:

```
morfologik  | tagset-converter --lang pl | parse | draw
```

The two translation engines supported by PSI-Toolkit use different tag-sets. The syntax-based statistical translator (Junczys, 2009) may be trained on the tag-set delivered by the default PSI-Toolkit lemmatizer and then needs no tag-set conversion. The rule-based translator works on its own tag-set. *tagset-converter* substitutes the tags delivered by the PSI-Toolkit lemmatizer with the tags used by the translator.

Users of the local version may specify their own tag-set converter by delivering a personalized set of tag-conversion rules. The *rules* option is used to specify the path to the tag-conversion text file.

The idea that stands behind tag-set conversion is to ensure that any type of annotator might be used within a PSI-pipeline. The annotations returned by an external tool are represented as PSI-lattice edges and the tag-set converter makes them applicable for other annotators.

2.8. Parsing

Parsing in PSI-Toolkit can be customized in two directions:

1. various types of parsers are admissible in the annotating process

2. the format of parsing annotation should be customizable to various needs.

The first postulate is satisified thanks to the PSI-lattice data structure. Thanks to the tag-converters a parser in a PSI-pipeline may work on an arbitrary set of tags. Currently PSI-Toolkit supports three different syntactic parsers, the first of them being the adaptation of an external tool:

link grammar parser for English (Sleator and Temperley, 1991)

shallow parser for Polish and French (Manicki, 2009)

deep parser for Polish (Graliński, 2006).

An exemplary output of the link parser displayed in a tree form is shown in Fig. 5. The deep parser returns the whole sentence structure as shown in Fig. 6.

Users may customize parsing by choosing the output format. Graphical output formats serve for educational purposes. Various textual formats facilitate further processing. See subsection 2.10. for details.

2.9. Translation Tools

PSI-Toolkit provides two MT engines: rule-based (named *Transferer*) and syntax-based statistical (named *Bonsai*). The rule-based PSI-Toolkit engine currently carries out translation from Polish into English and Spanish. Users of the local version may specify personal rules to execute rule-based translation between other languages, provided that the rules comply with the PSI-Toolkit format.

The statistical translator is trained on the European Parliament Proceedings Parallel Corpus (Koehn, 2005) and performs translation from Polish into English, French, Spanish or Italian. Users of the local version may customize the translator by providing their own translation rule set.

The *Bonsai* translator is a good example of applying several PSI-Toolkit annotators in one task. Fig. 7 shows an exemplary translation of a Polish sentence, whose English translation is: *I am sure that the European Union will solve their problems*, into Spanish. Although the final command is simple (*bonsai –lang pl –trg-lang es*), the translation process engages several PSI-Toolkit annotators. First, the translation model is trained on texts syntactically annotated by the PSI-Toolkit deep parser. Then, during run time, a source text is segmented, tokenized, lemmatized, tag-converted, and parsed syntactically. For each sentence in turn its parse trees are matched against the right-hand sides of translation rules in the translation model. Fig. 2.9. shows a subset of rules that potentially may be applied for the translation of a Polish structure *Jestem pewny, że VP (English: I am sure that VP)* (rule probabilities are omitted). The final translation of a sentence is obtained by recursive multiplying of translations for each parsed component of the sentence and choosing the translation that is estimated best by the target language model.

2.10. Output Formatting

The results of processing may be returned in various ways according to users' needs. For educational purposes graphical output seems most suitable. For engineering purposes more convenient formats are XML or JSON. Here is the list of the PSI-Toolkit writers:

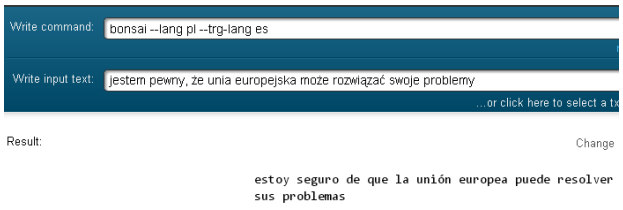

Figure 7: Bonsai - Syntax-based Statistical Translator

<VP>(0,8) –> Jestem pewny , że <VP>(4,8) :: estoy bastante seguro de que <VP>(4,8)
<VP>(0,8) –> Jestem pewny , że <VP>(4,8) :: estoy bastante seguro de que <FC>(4,8) en
<VP>(0,8) –> Jestem pewny , że <VP>(4,8) :: estoy seguro de que <VP>(4,8)
<VP>(0,8) –> Jestem pewny , że <VP>(4,8) :: estoy seguro de que <VP>(4,8) de
<VP>(0,8) –> Jestem pewny , że <VP>(4,8) :: estoy seguro de que la <VP>(4,8)
<VP>(0,8) –> Jestem pewny , że <VP>(4,8) :: estoy seguro de que la <VP>(4,8) de

Figure 8: A Subset of Translation Rules for Bonsai

bracketing-writer tags input text with square brackets (e.g.

```
(NP[AP[very large] house]
```

or with XML tags (e.g.

```
<np><ap>very large</ap> house</np>)
```

dot-writer presents results in a form of a directed graph, described in the *DOT* language used by the *GraphViz* software

gv-writer presents the results in a simple graphical form (directed graph) using *GraphViz* library

json-writer returns *JSON* output

perl-simple-writer creates a Perl array (used in Perl bindings)

psi-writer displays the content in the PSI format, used for representing the whole PSI-lattice

simple-writer prints the content of the lattice in a simple, human-readable way

2.11. Embedding PSI-Toolkit into an Application

PSI-Toolkit processors can be embedded in Perl or Python applications. This is an example that shows how to run the tokenizer inside a Python application:

```
import PSIToolkit
text = 'A short text to lemmatize'
command = 'tokenize --lang en | lemmatize'
psi = PSIToolkit.PipeRunner(command)
result = psi.run(text)
```

3. Anyone Can Build Their Own PSI-Toolkit Service

PSI-Toolkit is distributed in packages for most up-to-date Linux distributions. Two Linux applications are delivered in each distribution: *psi-pipe* and *psi-service*.
The former serves for using PSI-tools locally.
The latter allows users to design their own PSI-Toolkit web services. A service owner may set the PSI-Toolkit to their liking: tokenizers and segmenters may operate on SRX rule sets different from original ones, lemmatizers and POS-taggers may use their own lexicons and tagsets, translators may operate on translation rules delivered or supplemented by the owner.

4. Conclusions

The paper reports on the rationale standing behind PSI-Toolkit, an NLP set of tools designed in Poznań, Poland. The toolkit provides accesss to a set of NLP tools that deal mostly with the Polish language. The main idea of the solution is its open architecture. The toolkit can process texts in any natural language (provided it is UTF8-coded). The tools may be run with several options to differentiate the format of input and output. The rules used by PSI-Tools may be supplemented or replaced by the user. External tools may be incorporated into the PSI-Toolkit environment. Corpora annotated by other toolkits may be read into the PSI-lattice and further processed. Annotators using different tag-sets may co-operate in harmony.
The PSI-Toolkit package contains the *psi-service* application, which allows for free-license setting of a new web service (possibly with customized rule files). This feature will hopefully give rise to new instances of PSI-Toolkit web services.
Wikipedia lists 43 most popular NLP toolkits. One may expect the number to grow up in near future. Is it possible for a new NLP toolkit to be compatible with existing ones? The paper shows a positive example.

5. References

S. Bird, E. Klein, E. Loper. 2009. Natural Language Processing with Python, O'Reilly Media

F. Graliński, K. Jassem, M. Junczys-Dowmunt. 2012. PSI-toolkit: A Natural Language Processing Pipeline. In: A. Przepiorkowski, M. Piasecki, K. Jassem and P. Fuglewicz, editors *Computational Linguistics - Applications. Studies in Computational Inteeligence, Vol 458*, Heidelberg. Springer.

F. Graliński. 2007. Formalizacja nieciągłości zdań przy zastosowaniu rozszerzonej gramatyki bezkontekstowej, PhD thesis, Poznań, AMU.

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

M. Junczys-Dowmunt. 2009. It's all about the Trees - Towards a Hybrid Machine Translation System. In: *Proceedings of 4th International Multiconference on Computer Science and Information Technology, Mrągowo 2009*, pages 219-226.

P. Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In: *Conference Proceedings: the tenth Machine Translation Summit*, pages 79-86.

L. Manicki 2009. Płytki parser języka polskiego (Eng: A shallow parser for Polish), MSc thesis, Poznań, AMU.

T. Obrębski, M. Stolarski. 2005 UAM Text Tools - A text processing toolkit for Polish. In: *Proceedings of 2nd*

Language and Technology Conference, Poznań, 2005, pages 301-304.

A. Przepiórkowski, M. Bańko, R. Górski, B. Tomaszczyk 2012. Narodowy Korpus Języka Polskiego, Warsaw, Wydawnictwo Naukowe PWN.

A. Przepiórkowski, P. Bański. 2011. XML Text Interchange Format in the National Corpus of Polish. In: S. Goźdź-Roszkowski, editor, *Explorations across Languages and Corpora: PALC 2009*, pages 55-65, Frankfurt am Main, Peter Lang.

D. Sleator, D. Temperley. 1991. Parsing English with a Link Grammar, Carnegie Mellon University Computer Science technical report CMU-CS-91-196.