An Example of a Compatible NLP Toolkit

Krzysztof Jassem and Roman Grundkiewicz

Adam Mickiewicz University
ul. Wieniawskiego 1, 61-712 Poznań, Poland
{jassem,romang}@amu.edu.pl

Abstract. The paper describes an open-source set of linguistic tools, whose distinctive features are its customisability and compatibility with other NLP toolkits: texts in various natural languages and character encodings may be read from a number of popular data formats; all annotation tools may be run with several options to differentiate the format of input and output; rule lists used by individual tools may be supplemented or replaced by the user; external tools (including NLP tools designed in independent research centres) may be incorporated into the toolkit's environment.

Keywords: PSI-Toolkit, NLP tools, Polish language, software architecture, open source

1 Introduction

PSI-Toolkit is a set of NLP tools designed in the years 2011–2013 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 abbreviation for the Polish name of the laboratory) at Adam Mickiewicz University in Poznań, Poland. The secondary goal is to enable incorporation of tools developed at other NLP centres. For these reasons, the architecture of the toolkit is designed to allow for customisability and compatibility with other NLP toolkits.

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 (i.e. each NLP tool) of the processing pipeline adds new edges to the existing structure (see [6] for more details).

PSI-Toolkit may be personalised according to users' needs. Users may customise PSI-Toolkit in three ways: by selecting particular annotators to be used in the processing pipeline, by specifying run options for each annotator or by substituting annotation rules. Furthermore, users can easily combine tools of PSI-Toolkit with external applications in one processing pipeline using Unix shell.

The functionality of PSI-Toolkit attempts to combine selected features of well-known NLP toolkits. We follow the Stanford Natural Language Processing Group[1] in letting a user run PSI-Toolkit from a command line. Just like

[1] http://nlp.stanford.edu

in the NLTK toolkit [2] and UIMA[2] [14] we want programmers to be capable of building programs that call PSI-Toolkit annotators. We would like users to apply annotator pipelines, as e.g. in GATE[3]. Finally, we aim at encouraging pure linguists to use PSI-Toolkit, by delivering a friendly web-service — this is motivated by Apertium[4] [3].

The full range of PSI-Toolkit functionality is offered in the command-line mode, but the majority of functions are also available in the web-service (that can also be run locally). Fig. 1 shows the main window of the service.

Fig. 1. PSI-Toolkit web-service available at `http://psi-toolkit.amu.edu.pl`.

A standard PSI-Toolkit command is formed as a pipeline of annotators[5], e.g. `reader | segmenter | tokenizer | lemmatizer | writer`. In Sections 2, 3 and 4 we report on customizing various annotators: readers, processors and writers respectively. Then, we describe the different ways of using PSI-Toolkit in Section 5, and finally we draw conclusions in Section 6.

2 Reading Various Input Data

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 extracted text fragments. A user may customise 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:

[2] `http://uima.apache.org`
[3] `http://gate.ac.uk/`
[4] `http://www.apertium.org`
[5] The complete list of annotators can be found at `http://psi-toolkit.amu.edu.pl/help/documentation.html`

- `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 text from the Polish National Corpus XML files in TEI format [12],
- `utt-reader` — reads files in the UTT (UAM Text Tools) format[6],
- `psi-reader` — reads files in the PSI-Toolkit format[7].

A user may decide not to define the reader, leaving the task to the PSI-Toolkit. In that case, a special processor called `guessing-reader` guesses the input format. It works for most supported textual and binary formats.

Readers may be run with various options depending on the type of the reader: `txt-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, incuding compressed XML-based file formats, such as PPTX or XLSX. It is also possible to write custom rules for parsing a user's own XML-based format.

Currently PSI-Toolkit supports reading of two annotation formats designed externally: `nkjp-reader` processes the TEI encoding format [1] that contains linguistically annotated sentences of the NKJP corpus [11], and `utt-reader` reads a specialised format that results from annotating text with the UTT tools [10]. 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 tokenised 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.

2.1 Processing the Language of the Text

PSI-Toolkit puts no constraints on the language of the text. It recognises the input encoded in one of the 40 most popular character encoding standards and can automatically convert the text to UTF-8. If no language is specified, a special processor, called `lang-guesser`, tries to recognise 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. There, `lang-guesser` creates an edge tagged with the language code `!en`[8], 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 customise the PSI-Toolkit annotators to process the text according to rules specific for a language. For example, setting the `--lang` option to `en`

[6] `http://http://utt.amu.edu.pl/files/utt.html`
[7] `http://psi-toolkit.wmi.amu.edu.pl/help/psi-format.html`
[8] Tags that begin with an exclamation mark are so-called plane tags [6].

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.

Fig. 2. Extraction of a fragment written in a specified language.

makes the `segmenter` use the sentence-splitting rules specific for the English language.

3 Customizing PSI-Toolkit Processors

This section reports on customising processors, i.e. annotators that add edges to the PSI-Toolkit lattice. These include spell-checkers, segmenters, taggers, parsers and translators.

3.1 Spell-Checking

Spell-checking is customised by choosing the language of the input text. Currently, PSI-Toolkit uses the GNU Aspell spell checker[9] and applies its lexicons that support over 80 languages. Each `aspell` suggestion for an unrecognised token is converted into a PSI-lattice edge spanning over the token. A user of the local version of the toolkit may personalise spell-checking by adding new lexicons either for supported or unsupported languages. Fig. 3 shows a PSI-Toolkit use-case for a spell-checker.

3.2 Tokenization

`Tokenizer` splits texts into tokens according to rules defined in an SRX (Segmentation Rules eXchange[10]) file. PSI-Toolkit supports segmentation for 9 languages (and one default language). Tokenization may be customised by choosing the language and/or the maximum length of the token. A user of the local version may deliver a personalised SRX file (this is done via the `--rules` option of `tokenizer`) either for supported or unsupported languages.

[9] `http://aspell.net/`
[10] `http://www.gala-global.org/oscarStandards/srx/srx20.html`

Command
`tokenize | aspell --lang en`

Input
I enjoy travleling

Output
I
enjoy
travleling—travelling—traveling—travailing—travellings—ravelling

Fig. 3. A use-case for a spell-checker.

3.3 Sentence-Splitting

`Segmenter` splits texts into segments (i.e. sentences) according to rules defined in an SRX file. PSI-Toolkit supports segmentation for 9 natural languages: Polish, English, German, Italian, French, Spanish, Finnish, Turkish and Russian. For unsupported languages a "default" is assumed. This triggers the most general segmentation rules that work satisfactory for most Indo-European languages. Sentence-splitting may be customised by choosing the language and/or the maximum length of the sentence.

A user of the local version may deliver a personalised SRX file (this is done via the `--rules` option of `segmenter`) either for supported or unsupported languages.

3.4 Lemmatization

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

1. the lexicon (in the binary or plain text format),
2. the text file containing part-of-speech information,
3. the text file containing morphological information.

3.5 Tag-Set Conversion

PSI-Toolkit puts no constraints on the format of the information returned by the lemmatiser. It is allowed for different tools called in the same pipeline to operate on different tag-sets. For example, the Polish lemmatiser supported by PSI-Toolkit is based on the `morfologik`[11] tag-set, whereas the deep parser operates on a different tagset developed for Tree-generating Binary Grammar [5]. Still, the two tools may be called in one pipeline using a tag-set converter that maps

[11] `http://sourceforge.net/projects/morfologik`

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 3.7 use different tag-sets. The syntax-based statistical translator [7] may be trained on the tag-set delivered by the default PSI-Toolkit lemmatiser and then needs no tag-set conversion. The rule-based translator, however, works on its own tag-set. `Tagset-converter` substitutes the tags delivered by the PSI-Toolkit lemmatiser with the tags used by the translator.

Users of the local version may specify their own tag-set converter by delivering a personalised 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.

3.6 Parsing

Parsing in PSI-Toolkit can be customised in two ways: various types of parsers are admissible in the annotating process, and the format of parsing annotation is customisable 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 [13],
- shallow parser for Polish and French [9],
- deep parser for Polish [4].

An exemplary output of the link parser displayed in a tree form is shown in Fig. 4. The deep parser returns the whole sentence structure.

Users may customise parsing by choosing the output format. Graphical output formats serve for educational purposes. Various textual formats facilitate further processing. See Section 4 for details.

3.7 Translation Tools

PSI-Toolkit provides two machine translation 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.

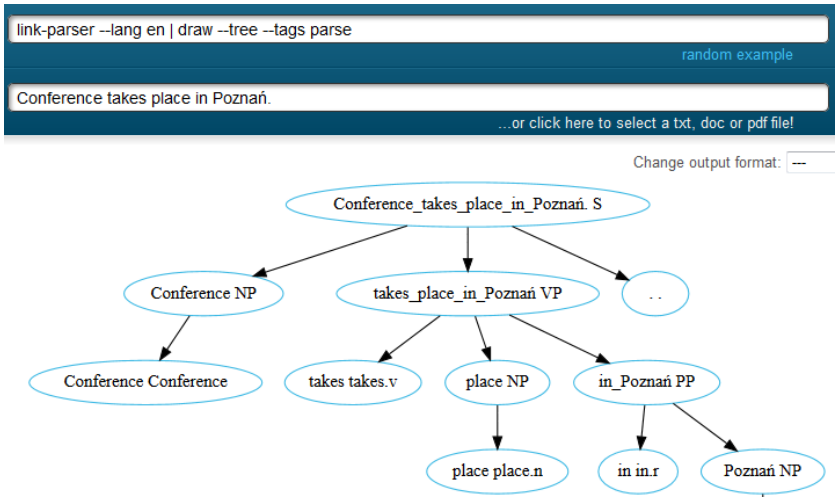

Fig. 4. Link Grammar adapted in PSI-Toolkit.

The statistical translator is trained on the European Parliament Proceedings Parallel Corpus [8] and performs translation from Polish into English, French, Spanish or Italian. Users of the local version may customise the translator by providing their own translation model in a format of textual translation rules (see Fig. 6).

The `bonsai` translator is a good example of applying several PSI-Toolkit annotators in one task. Fig. 5 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.

Command
`bonsai --lang pl --trg-lang es`

Input
jestem pewny, że unia europejska może rozwiązać swoje problemy

Output
estoy seguro de que la unión europea puede resolver sus problemas

Fig. 5. Bonsai — Syntax-based Statistical Translator.

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, tokenised, lemmatised, 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. 6 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.

```
<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)
```

Fig. 6. A Subset of Translation Rules for Bonsai.

4 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:

- `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.

5 PSI-Toolkit Usage

PSI-Toolkit is distributed in packages for the most popular Linux distributions. Two applications are delivered in each distribution: `psi-pipe` and `psi-service`.

The former serves for using annotators locally as a shell command. The user may easily supplement the PSI-Toolkit pipeline by Unix text utils.

The latter allows users to design their own PSI-Toolkit web services. A service owner may set the PSI-Toolkit to their liking: XML reader may use custom parsing rules, tokenisers and segmenters may operate on SRX rule sets different from original ones, lemmatisers and POS-taggers may use their own lexicons and tagsets, translators may operate on translation rules delivered or supplemented by the owner.

Moreover, PSI-Toolkit processors can be embedded in Perl or Python applications. This is an example that shows how to run the tokeniser 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)
```

6 Conclusions

The paper reports on the rationale standing behind PSI-Toolkit, a customisable and compatible set of linguistic tools. The toolkit provides accesss to a set of NLP tools that deal mostly with Polish, but also (with lesser extent) with English, French, German, Spanish and Russian.

The main idea of the solution is its extensible architecture that allows for compatibility with other NLP toolkits. The toolkit can process texts in a number of natural languages and character encodings. Input data can be read from various data formats, including formats generated by other toolkits. The tools may be run with several options in order to differentiate the format of input and output. The rules used by individual tools may be supplemented or replaced by the user. Annotators using different tag-sets may co-operate in harmony. Finally, external tools may be incorporated into the PSI-Toolkit environment, and the PSI-Toolkit itself may be a part of Perl or Python application.

The PSI-Toolkit package contains the `psi-service` application, which allows for free-license setting of a new web service. This feature will hopefully give rise to new instances of PSI-Toolkit web services.

Wikipedia lists 43 most popular NLP toolkits[12]. 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.

[12] `https://en.wikipedia.org/wiki/Outline_of_natural_language_processing`, access: 8 August, 2015.

References

1. Bański, P., Przepiórkowski, A.: The tei and the ncp: the model and its application. In: LREC2010 Workshop on Language Resources: From Storyboard to Sustainability and LR Lifecycle Management. ELRA, Valletta, Malta (2010)
2. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media, Inc., 1st edn. (2009)
3. Forcada, M.L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J.A., Sánchez-Martínez, F., Ramírez-Sánchez, G., Tyers, F.M.: Apertium: A free/open-source platform for rule-based machine translation. Machine Translation 25(2), 127–144 (2011)
4. Graliński, F.: Some methods of describing discontinuity in Polish and their cost-effectiveness. Lecture Notes in Artificial Intelligence 4188, 69–77 (2006)
5. Graliński, F.: Formalizacja nieciągłości zdań przy zastosowaniu rozszerzonej gramatyki bezkontekstowej. Ph.D. thesis, Adam Mickiewicz University in Poznań, The Faculty of Mathematics and Computer Science, Poznań (2007), supervisor: Zygmunt Vetulani
6. Graliński, F., Jassem, K., Junczys-Dowmunt, M.: PSI-Toolkit: Natural language processing pipeline. Computational Linguistics — Applications 458, 27–39 (2012)
7. Junczys-Dowmunt, M.: It's all about the trees — towards a hybrid syntax-based MT system. In: 4th International Multiconference on Computer Science and Information Technology. pp. 219–226. Mrgowo, Poland (2009)
8. Koehn, P.: Europarl: A parallel corpus for statistical machine translation. In: Conference Proceedings: the tenth Machine Translation Summit. vol. 5, pp. 79–86 (2005)
9. Manicki, L.: Płytki parser języka polskiego (eng: A shallow parser for polish) (2009), supervisor: Krzysztof Jassem
10. Obrębski, T., Stolarski, M.: UAM Text Tools — a text processing toolkit for polish. Proceedings of 2nd Language and Technology Conference pp. 301–304 (2005)
11. Przepiórkowski, A., Bańko, M., Górski, R., Barbara, L.T. (eds.): Narodowy Korpus Języka Polskiego. Wydawnictwo Naukowe PWN (2012)
12. Przepiórkowski, A., Bański, P.: Xml text interchange format in the national corpus of polish. The proceedings of Practical Applications in Language and Computers PALC pp. 55–65 (2009)
13. Sleator, D.D., Temperley, D.: Parsing english with a link grammar. Tech. rep., Carnegie Mellon University Computer Science technical report CMU-CS-91-196 (1995)
14. Verspoor, K., Baumgartner Jr, W., Roeder, C., Hunter, L.: Abstracting the types away from a uima type system. From Form to Meaning: Processing Texts Automatically pp. 249–256 (2009)