PSI-Toolkit – how to turn a linguist into a computational linguist

Krzysztof Jassem

Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland

Abstract. The paper presents PSI-Toolkit, a set of text processing tools, being developed within a project funded by the Polish Ministry of Science and Higher Education. The toolkit serves two objectives: to deliver a set of advanced text processing tools (with the focus set on the Polish language) for experienced language engineers and to help linguists without any technological background learn using linguistics toolkits. The paper describes how the second objective can be achieved: First, a linguist, thanks to PSI-Toolkit, becomes a conscious user of NLP tools. Next, he designs his own NLP applications.

Keywords: Tagging, Classification and Parsing of Text, NLP Toolkits

1 Introduction

Our experience shows that most computational linguists are computer scientists, who have learned linguistics as their second major field of study, rather than the other way round. We are of the opinion that so few *pure* linguists become computational linguists because NLP tools are presented to them in an unappealing way. Let us present a few examples that may support this thesis.

1.1 The Stanford Natural Language Processing Group Toolkit

Stanford CoreNLP [3] is a suite of NLP tools designed for the analysis of raw English texts. "The goal of the project is to enable people to quickly and painlessly get complete linguistic annotations of natural language texts." However, the first sentence explaining the usage of the toolkit reads: "Before using Stanford CoreNLP, it is usual to create a configuration file (a Java Properties file)." If a linguist is not discouraged by the *Java Properties file*, here what comes next: In particular, to process the included sample file input.txt you can use this command in the distribution directory: *Stanford CorNLP Command*

```
java -cp stanford-corenlp-2012-01-08.jar:stanford-corenlp-2011-12-27-models.jar:xom.jar:joda-time.jar-Xmx3g
edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokenize,ssplit,pos,lemma,ner,parse,dcoref -file input.txt
```

The output of the Stanford processor is an XML file - a format more suitable for machine analysis rather than for human reading. A drawback of Stanford CoreNLP from the point of view of language engineers is that the toolkit is licensed under the General Public License, which does not allow for using the code in the proprietary software.

1.2 UIMA Project

"Unstructured Information Management applications are software systems that analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user" ([4]). The tools (annotators) in the UIMA Project are available only as Java source codes and need compilation under a Java development environment. A potential user needs at least a preliminary course in Java programming in order to benefit from the UIMA project.

1.3 Teaching Basic Programming to Linguists

An attempt to use Stanford CoreNLP, UIMA or similar toolkits may tempt a linguist to learn basic programming skills. One of the manuals intended for linguists is Martin Wieser's [2]. The introduction sounds encouraging: "This book is mainly intended as an introduction to programming for linguists without any prior programming experience". The author of this paper tried to follow this approach with a group of bright students at interdisciplinary PhD studies: Language, Society, Technology and Cognition funded by the European Social Fund ([5]). At the end of the 16-hour course students had the skills to write Perl programs that could lemmatize, form frequency lists or concordance lists. When the course and the exam were over, the students asked why they should write programs like those instead of using existing tools. The argument that existing tools may not always satisfy their specific needs, did not seem convincing to them.

1.4 Natural Language Toolkit (NLTK)

Natural Language Toolkit [6] is geared towards less experienced users. The user downloads three files and soon is offered valuable results of text processing. There are two small "buts": 1) NLTK basic tools work on texts delivered by the authors or textual files, which first have to be converted to the NLTK format. 2) The Python GUI is in fact a form of the command line (not a graphical interface a linguist is used to work with).

1.5 General Architecture for Language Engineering (GATE)

GATE ([7]) is one of the most mature NLP toolkits, dating from 1995. The system is intended for both language engineers who can develop their programs including GATE modules and for linguists who can write their own grammars for GATE tools. A linguist can process his own (set of) documents (of various formats). However, in order to obtain the first annotation, the user must overcome a few difficulties. First, he has to load a CREOLE plugin. Then, it is required to create processing resources. Next, an application has to be set up from the processing resources. The application is not likely to work, as two hardly obvious conditions must be fulfilled: 1) Each process forming the application must be assigned to a specific document, 2) The pipelines of processes must be "logical".

Contrary to reasonable expectation the user will not see the immediate result of processing displayed on the screen, but has to go to *Language resources* to be able to view them.

1.6 Apertium

Apertium ([8]) is a free open-source machine translation platform that provides an engine for using existing machine translation systems as well as building new ones. Two features make the toolkit attractive for linguists: 1) the translation window, thanks to which a user may easily and immediately obtain the result of translation, 2) user-friendly means of creating self-developed systems (it suffices to edit a few dictionary files and a rule file to design an MT system).
We would like to take the Apertium approach in PSI-Toolkit. However, we would like to expand the domain of the toolkit so that it could assist in various linguistic tasks, not only Machine Translation.

2 PSI-Toolkit

2.1 PSI-Toolkit outline

PSI-Toolkit is a set of tools designed for text processing, released under Lesser General Public License. This is a free license, which, in addition to GPL, could be linked with proprietary software and further distributed under any terms. The toolkit is intended for both language engineers and pure linguists (without computer education). The former group of users should be satisfied with Java, Perl and Python libraries, whereas the latter group is encouraged to use PSI-Toolkit by means of a user-friendly web portal.

2.2 PSI-Toolkit processors

The programs included in the PSI-Toolkit are called PSI-processors. There are three types of PSI-processors: readers, annotators and writers. Readers process the text (input either from keyboard or from a file) in order to initialize the main data structure, a so-called PSI-lattice (see [1] for the description of the PSI-lattice). Annotators (e.g. a tokenizer, a lemmatizer, a parser) add new edges to the PSI-lattice. Writers convert the PSI-lattice to a graphical or textual format and re-direct the result to the output device.

PSI-Toolkit readers PSI-Toolkit readers read all formats of files listed in [9] and additionally .pdf files. A reader initializes the PSI-Lattice by converting individual characters of the textual content into vertices of the lattice. For instance, in the web version of the toolkit, the text typed into the edit window is processed by *txt-reader*.

PSI-Toolkit annotators PSI-Toolkit annotators are core processes in the system. The current version supports, among others, a tokenizer, a sentence-splitter, a lemmatizer and a shallow parser. A unique annotator is a bilexicon processor, which returns all the equivalents of all possible translations of input lemmas.

PSI-Toolkit writers PSI-Toolkit writers are a distinguishing feature of the project. There are two types of PSI writers: textual and graphical. The former return a textual representation of the PSI-lattice, whereas the latter display its visual representation. Both types can be customized by a user, who can filter the labels of edges to be returned. Also, the output of the textual writer may be simplified by filtering the information attached to edges.

2.3 PSI-Toolkit web portal

The PSI-Toolkit portal is accessible via a web browser at [10]. Fig. 1 shows an example of using the toolkit in a web window.

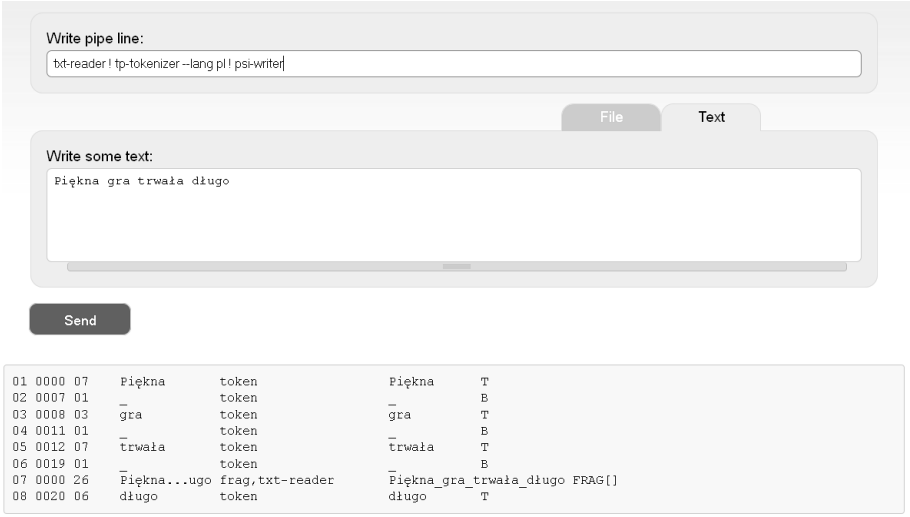

Fig. 1. PSI-Toolkit web portal

The user types a text into the edit window, e.g. *Piękna gra trwała długo. (Eng. A beautiful game lasted long.)* and specifies a command: a sequence of processors to execute. (This is facilitated by a list of prompts: most often used commands.) The processors are run in the order specified in the command. The exemplary sentence consists of ambiguous words (word ambiguity in the Polish language is an irregular phenomenon). *Piękna* is the feminine form of the adjective *piękny* or the genitive form of the noun *piękno (Eng. beauty)*. *Gra* is the base form of the noun *gra (Eng. game)* or a form of the verb *grać (Eng. to*

play). Trwała is the past form of the verb *trwać (Eng. to last)* or the feminine form of the adjective *trwały (Eng. long-lasting)*, or the base form of the noun *trwała (Eng. durable haircut)*. The only unambiguous word, the adverb *długo (Eng. long)* determines the syntactic interpretation of the whole expression. The PSI output lists each edge of the PSI-lattice in a separate line. In Fig. 1: Line 01 corresponds to the edge spanning over the first 6 characters of the input (start position is equal to 0000, offset counted in bytes is equal to 0007, as *ę* is represented by two bytes). Line 02 describes the space between the first and the second token of the input. Line 07 corresponds to the edge spanning over the whole input (the edge has been constructed by *txt-reader*).

2.4 Linux distribution

PSI-Toolkit is also distributed in the form of two Linux binaries: PSI-Pipe and PSI-Server. PSI-Pipe may be installed and used on personal computers, whereas PSI-server allows for the creation of other PSI-Toolkit web pages.

2.5 PSI pipeline

The PSI-toolkit command is specified as a pipeline of processors. If PSI-Toolkit is used under Linux on a personal computer, the processors should be invoked in a bash-like manner. For example, in order to process the string *Piękna gra trwała długo* in the way equivalent to that shown in Figure 1, the following pipeline should be formed:

```
>echo 'Piękna gra trwała długo' |
psi_pipe ! txt_reader ! tp-tokenizer --lang pl ! psi-writer
```

Interestingly this approach allows for using standard Linux commands e.g. *sort*, *grep* besides PSI pipelines. Moreover, PSI processors may be replaced or supplemented by external tools in the PSI pipeline. The PSI engine will add annotations provided by external tools to the PSI-lattice. It is admissible to use two different processors of the same type in the same pipeline. For example, running two different sentence splitters in the same process (in any order) may result in two different sentence splits. The ambiguity is stored in the PSI-lattice (it may or may not be resolved in further processing).

2.6 Switches

The use of the PSI processor may be customized be means of switches (options). Table 1 shows the list of admissible switches for *simple-writer*, the processor for printing the PSI-lattice in a simple, human-readable way.

2.7 Examples of usage

A combination of processors may result in an output format customized to the user's needs. Table 2 shows the results printed by *simple-writer* customized to show respectively: a) only tokens, b) only lemmas or c) only lexemes.

Table 1. Use of switches for *simple-writer*

–sep arg	set separator between basic edges
–alt-sep arg	set separator between alternative edges
–linear	skip cross-edges
–no-alts	skip alternative edges
–with-blank	do not skip edges for 'blank' characters
–tag arg	filter edge tags

Table 2. Output of *simple-writer* depends on tag filtering

Piękna	piękno|piękny	piękno+subst|piękny+adj
gra	gra|grać	gra+subs|grać+verb
trwała	trwać|trwała|trwały	trwać+verb|trwała+subst|trwały+adj
długo	długo	długo+adv
a) tokens	b) lemmas	c) lexemes

Table 3 shows the output if the pipeline contains the *bilexicon* command.

By the end of the project a combination of PSI-processors will deliver on-line translation.

The visual writer displays either the whole lattice built by the processors or a part of it, if tag-filtered.

Fig. 2 shows the whole lattice after the lemmatization .

Fig. 3 displays the lattice without character vertices.

2.8 Java, Perl, Python libraries

PSI tools are also accessible as libraries of selected programming languages. The user may include the PSI tools as Java, Perl or Python modules.

2.9 Natural languages processed by PSI-Toolkit

There are no restrictions on languages analyzed by PSI processors (UTF-8 is used). One PSI pipeline may consist of processors defined for various languages. However, the tools delivered by the authors are oriented mainly towards the Polish language (some processors are also defined for English). The authors hope that PSI-Toolkit will bring together the Polish language processing tools, which are dispersed (see [11] for an exhaustive list of NLP tools and resources for Polish). The proof of this concept has been implemented on the morphosyntactical

Table 3. Output of *simple-writer* depends on tag filtering

beauty+subst|beautiful+adj|excellent+adj
game+subst|play+verb
last+verb|stay+verb|abide+verb|permanent+adj|durable+adj|lasting+adj
long+adv

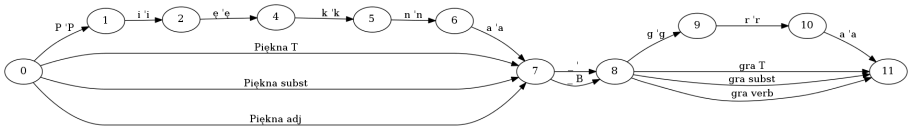

Fig. 2. Output of the PSI graphical writer: PSI-lattice after lemmatization

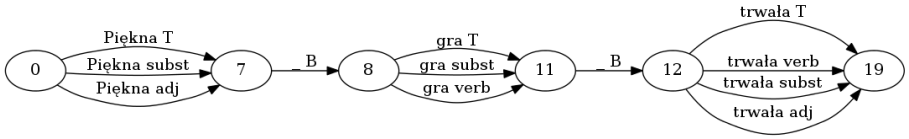

Fig. 3. Output of the PSI graphical writer: PSI-lattice without character vertices

level: the external morphological lemmatizer *Morfologik* ([12]) has been incorporated into the PSI toolkit and can be run in the PSI pipeline instead of or besides (!) the PSI-dedicated lemmatizer.

3 PSI-Toolkit as a didactic tool for linguists

This section describes one of the objectives for the creating PSI-Toolkit: *turning a linguist into a computational linguist*. The training path involves building computer skills of a *pure* linguist, by means of PSI-Toolkit:

1. With the aid of PSI-Toolkit perform some basic linguistic tasks (e.g. tokenize, sentence split, lemmatize, parse)
   - Show how syntactical parsing may solve word disambiguities
   - Introduce PSI-writers in the appropriate order:
     - Use the graphical writer (*gv-writer*) first
     - Then proceed with *simple-writer*
     - Finally use the full *psi-writer*
2. Introduce the 'bash' shell
   - Teach the basics of the 'bash' shell
   - Define simple exercises that require sorting or greping
   - Teach the usage of the 'grep' and 'sort' commands
3. Use PSI-Toolkit in the 'bash' shell
   - Download the PSI-Tools package from the PSI portal
   - Install the package
   - Define PSI pipelines including 'bash' commands
4. Introduce a "linguistic" programming language e.g. Perl or Python
   - Define simple tasks that cannot be handled with the knowledge acquired so far (e.g. generating a text index or a frequency list)
   - Teach the basics of the language until students are able solve the above problems
   - Suggest using PSI-Toolkit libraries in the student's applications

5. Design your own toolkit portal
   Download the PSI–Server package
   Publish your own PSI-Toolkit web-page
6. Introduce other NLP toolkits such as: NLTK, UIMA.

4 Conclusions

The paper presents the main ideas and the architecture of PSI-Toolkit, an open source NLP toolkit. The PSI tools may be accessed via a web browser or run locally. A PSI processing command should be formed as the pipeline, which may also include external tools. The PSI wrappers make it possible to use the PSI tools in Java, Perl or Python applications.
The toolkit is being developed within a project funded by the Polish Ministry of Science and Higher Education, which runs from April 2011 to April 2013. The progress of the project may be traced systematically at the Psi-Toolkit portal. At the moment this paper was written the portal provided 17 NLP processors. Providing tools for deep parsing and Machine Translation is scheduled for the months to come.
One of the main objectives underlying the creation of the toolkit was to encourage pure linguists to use and create text processing tools. The paper presents a path from a linguist to a computational linguist or even a language engineer. The incentive is a user-friendly web application, which returns comprehensive results for users without any programming background. Further steps to take depend on the increasing needs of a linguist already convinced that computer education could be easy and useful.

References

1. Jassem, K. Gralinski, F., Junczys-Dowmunt, M.: PSI-toolkit: A Natural Language Processing Pipeline. Computational Linguistics - Application. Springer, Heidelberg (to appear)
2. Wiesser, M.: Essential progamming for lingusts. Edinburgh University Press Ltd, Edinburgh (2009)
3. The Stanford Natural Language Processing Group `http://nlp.stanford.edu/software/corenlp.shtml`
4. Apache UIMA `http://uima.apache.org/`
5. LSTC `http://unikat.amu.edu.pl/studia/lstc/eng`
6. NLTK `http://www.nltk.org/`
7. GATE `http://gate.ac.uk/`
8. Apertium `www.apertium.org`
9. Apertium format handling `http://wiki.apertium.org/wiki/Format_handling`
10. PSI-Toolkit portal `psi-toolkit.wmi.amu.edu.pl`
11. CLIP `http://clip.ipipan.waw.pl/`
12. Morfologik `http://morfologik.blogspot.com/`