A Web Service Implementation of Linguistic Annotation for Slovene and English

Senja Pollak[1,2], Nejc Trdin[1,3], Anže Vavpetič[1,3] and Tomaž Erjavec[1,3]
[1]Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia
[2]Faculty of Arts, Aškerčeva 2, 1000 Ljubljana
[3]International Postgraduate School Jožef Stefan, Jamova cesta 39, 1000 Ljubljana, Slovenia
{senja.pollak, nejc.trdin, anze.vavpetic, tomaz.erjavec}@ijs.si

Abstract

This paper presents a web service for automatic linguistic annotation of Slovene and English texts. The texts are tokenised, morphosyntactically tagged and lemmatised by the ToTrTaLe annotation tool, while the web service for this annotation is made available in the Orange4WS and the ClowdFlows workflow construction environments. The workflows enable the users to apply the annotation tool as an elementary constituent for other natural language processing workflows. The user can upload the text(s) in different formats (TXT, DOC, DOCX, PDF, ZIP), convert them to plain text and annotate them with ToTrTaLe. The paper also proposes several improvements of the ToTrTaLe tool based on the identification of various types of errors of the existing implementation, and implements these improvements as a post-processing step in the workflow.

Implementacija spletnega servisa za jezikoslovno označevanje slovenskega in angleškega jezika

Prispevek predstavi implementacijo spletnega servisa za jezikoslovno označevanje slovenskega in angleškega jezika. Program ToTrTaLe je kot spletni servis uporabnikom na voljo v okoljih za izgradnjo delotokov Orange4WS in ClowdFlows in omogoča tokenizacijo, oblikoskladenjsko označevanje in lematizacijo besedil. Delotoki omogočajo uporabniku, da uporabi jezikoslovno označevanje kot elementarni gradnik pri večjih delotokih za analizo besedil. Uporabnik lahko naloži besedila v različnih formatih (TXT, DOC, DOCX, PDF, ZIP), jih pretvori v navadno besedilo in jih označi s ToTrTaLe. Prispevek tudi predlaga več izboljšav za ToTrTaLe, ki temeljijo na identifikaciji različnih vrst napak obstoječe implementacije, in jih implementira kot dodaten korak delotoka.

1. Introduction

In corpus linguistics, part-of-speech tagging (PoS tagging), also called word-level grammatical tagging, is the process of marking up word tokens in a text (corpus) as corresponding to a particular part of speech, based on the lexicon giving the possible PoS tags of the word and the context in which the word appears. PoS-tagging algorithms fall into two groups: rule-based taggers and statistical taggers where the PoS tags are learned from a manually annotated text corpus. For languages with rich inflection, such as Slovene, it is better to speak of morphosyntactic annotations or descriptions (MSDs) rather than PoS tags, as such MSDs contain much more information than do PoS tags. For example, the PoS tagsets for English have typically from 20 – 60 different tags, while Slovene has over 1,000 MSDs.

This paper focuses on a particular tool for automatic morphosyntactic tagging, named ToTrTaLe (Erjavec et al., 2011). A brief description of ToTrTaLe is presented in Section 2. As one of the main contributions of this work is the implementation of ToTrTaLe as a web service which can be used as an ingredient of complex NLP workflows, we first motivate this work in Section 3 by a short introduction to web services, workflows and by presenting two specific workflow construction environments Orange4WS and ClowdFlows. The main contributions of this research are presented in Sections 4 and 5. Section 4 presents the implementation of the ToTrTaLe analyser as a web service in two service-oriented workflow construction and management platforms Orange4WS and ClowdFlows. Section 5 presents the proposed improvements of the ToTrTaLe tool based on the identification of several types of errors of the existing implementation. These error corrections are implemented as a part of our web-service.

2. The ToTrTaLe annotation tool

ToTaLe (Erjavec et al., 2005) is short for Tokenisation, Tagging and Lemmatisation and is the name of a script implementing a pipeline architecture comprising these three processing steps. While the tool makes some language specific assumption, they are rather broad, such as that text tokens are (typically) separated by space; otherwise, the tool itself is language independent and relies on external language resources. The tool is written in Perl and is reasonably fast. The greatest speed bottleneck is the tool start-up, mostly the result of the lemmatisation module, which for Slovene contains thousands of rules and exceptions.

In the context of the JOS project (Erjavec et al., 2010) the tool was re-trained for Slovene and made available as a Web application at http://nl.ijs.si/jos/analyse/. It allows pasting the text to be annotated into the form or uploading a plain-text UTF-8 file and either have the annotated text displayed or downloaded as a ZIP file.

The tool (although not the Web application) has been recently extended with another module, Transcription, and the new edition is called ToTrTaLe (Erjavec, 2011). The transcription step is used for modernising historical language (or, in fact, any non-standard language), and the tool was used as the first step in the annotation of a reference corpus of historical Slovene (Erjavec, 2012a). And additional extension of ToTrTaLe is the ability to process heavily annotated XML document conformant to the Text Encoding Initiative Guidelines (TEI, 2007).

The Web service presented in this paper uses To(Tr)TaLe with models for Slovene and English, but as the historical language models are not as mature as those for contemporary language, this extra functionality is not discussed here further. In the rest of this section we present the main modules of To(Tr)TaLe and also their models for Slovene and English.

2.1. The tokenisation module

The multilingual tokenisation module mlToken[1] is written in Perl and in addition to splitting the input string into tokens, it also assigns to each token its token type, e.g. XML tag, sentence final punctuation, digit, abbreviation, URL, etc. and preserves (subject to a flag) white-space, so that the input can be reconstituted from the output.

The tokeniser can be fine-tuned by putting punctuation into various classes (e.g. word-breaking vs. non-breaking) and also uses several language-dependent resource files: a list of abbreviations ("words" ending in period, which is a part of the token and does not necessarily end a sentence); a list of multi-word units (tokens consisting of several space-separated "words"); and a list of (right or left) clitics, i.e. cases where one "word" should be treated as several tokens. Such resource files allow for various options to be expressed, although not all, as will be discussed in section 5.

The tokenisation resources for Slovene and English were developed by hand, and cover most typical exceptions in both languages.

2.2. Tagging

For tagging words in the text with their context disambiguated PoS tags (or, better, morphosyntactic annotations) we use TnT (Brants, 2000), a fast and robust tri-gram tagger.

For Slovene, the tagger has been trained on jos1M, the 1 million word JOS corpus of contemporary Slovene (Erjavec et al., 2010), and is also given a large background lexicon extracted from the 600 million word FidaPLUS reference corpus of contemporary Slovene (Arhar Holdt and Gorjanc, 2007). The English model was trained on the MULTEXT-East corpus (Erjavec, 2012b), namely the novel "1984". This is of course a very small corpus, so the resulting model is not very good. However, it does have the advantage of using the MULTEXT-East tagset, which is compatible with the JOS one.

2.3. Lemmatisation

For lemmatisation To(Tr)TaLe uses CLOG (Erjavec and Džeroski, 2004), which implements a machine learning approach to the automatic lemmatisation of (unknown) words. CLOG learns on the basis of input examples (pairs word-form/lemma, where each morphosyntactic tag is learnt separately) a first-order decision list, essentially a sequence of if-then-else clauses, where the defined operation is string concatenation. The learnt structures are Prolog programs but in order to minimise interface issues we made a converter from the Prolog program into one in Perl.

The Slovene lemmatiser was trained on a lexicon extracted from the jos1M corpus, and the lemmatisation of contemporary language is reasonably accurate, with 92% on unknown words. However the learnt model, given that there are 2,000 separate classes, is quite large: the Perl rules have about 2MB, which makes loading the lemmatiser slow.

The English model was trained on the English MULTEXT-East corpus, which has about 15,000 lemmas and produces a reasonably good model, especially as English is fairly simple to lemmatise.

[1] mlToken was written in 2005 by Camelia Ignat, then working at the EU Joint Research Centre in Ispra, Italy.

3. Web services and workflows

A Web service is a method of communication between two electronic devices over the web. The W3C defines a Web service as "a software system designed to support interoperable machine-to-machine interaction over a network". A Web service's functionalities are described in a machine-processable format i.e., the Web Services Description Language, known by the acronym WSDL. Other systems interact with the Web service in a manner prescribed by its description using SOAP XML messages, typically conveyed using HTTP in conjunction with other Web-related standards. The W3C also states that we can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations, and arbitrary Web services in which the service may expose an arbitrary set of operations.

3.1. Workflow construction platforms

Main data mining environments that allow for workflow composition and execution, implementing the visual programming paradigm, include Weka (Witten et al., 2011), Orange (Demšar et al., 2004), KNIME (Berthold et al., 2007) and RapidMiner (Mierswa et al., 2006). The most important common feature is the implementation of a workflow canvas where workflows can be constructed using simple drag, drop and connect operations on the available components, implemented as graphical units named widgets. This feature makes the platforms suitable for use also by non-experts due to the representation of complex procedures as relatively simple sequences of elementary processing steps (workflow components implemented as widgets).

In this work, we use two recently developed service-oriented environments for data mining workflow construction and execution: Orange4WS and ClowdFlows.

The first platform Orange4WS (Podpečan et al., 2012) is distinguished from other main data mining platforms by its capacity of including web services into data mining workflows, allowing for distributed processing. Such a service-oriented architecture has already been employed in Taverna (Hull et al., 2006), a popular platform for biological workflow composition and execution. Using processing components implemented as web services enables remote execution, parallelisation, and high availability by default. A service-oriented architecture supports not only distributed processing but also distributed development.

The second platform ClowdFlows (Kranjc et al., 2012) is distinguished from other main data mining platforms by its capacity of workflow sharing. Sharing of workflows has previously been implemented through the myExperiment website of Taverna (Hull et al., 2006). This website allows the users to publicly upload their workflows so that they are made available to a wider audience. Furthermore, publishing a link to a certain workflow in a research paper allows for simpler dissemination of scientific results. However, the users who wish to view or execute these workflows are still

required to install the specific software in which the workflows were designed and implemented. On the other hand, the ClowdFlows platform implements the described features also with one major advantage. ClowdFlows requires no installation and can be run on any device with an internet connection, using any modern web browser. The ClowdFlows platform is described in more detail below.

3.2. The ClowdFlows platform

ClowdFlows is implemented as a cloud-based application that takes the processing load from the client's machine and moves it to remote servers where experiments can be run with or without user supervision. The user does not need to perform any specific installation. ClowdFlows consists of the workflow editor (the graphical user interface, as shown in Figure 1) and the server-side application which handles the execution of the workflows and hosts a number of publicly available workflows.

The workflow editor consists of a workflow canvas and a widget repository, where widgets represent embedded chunks of software code. The widgets are separated into categories for easier browsing and selection and the repository includes a wide range of readily available widgets. Our NLP processing modules have also been implemented as such widgets.

By using ClowdFlows we were able to make our NLP workflow public, so that anyone can use and execute it. The workflow is exposed by a unique URL, which can be accessed from any modern Web browser. Whenever the user opens a public workflow, a copy of this workflow appears in her private workflow repository. The user can execute the workflow and view its results or expand it by adding or removing widgets. Any user can therefore use ToTrTaLe as a pre-processing step in any other NLP workflow.

4. ToTrTaLe web service implementation

In this section we present the services we implemented and also some details regarding the implementations. All services were implemented in the Python programming language, using Orange4WS API and additional freeware software packages. Services are currently adapted to run on Unix-like operation systems, but are easily transferable to other operation systems.

4.1. Implemented web services

We implemented two web services, which constitute the main implementation part of this work. The first service converts the files to plain text. The second service uses ToTrTaLe to annotate the texts.

4.1.1. Converting input data

The first service parses the input data and converts it into plain text. The input corpus can be uploaded in various formats, either as a single file or as several files compressed in a ZIP file. The supported formats are PDF, DOC, DOCX, TXT and HTML, the latter passed to the service in the form of an URL.

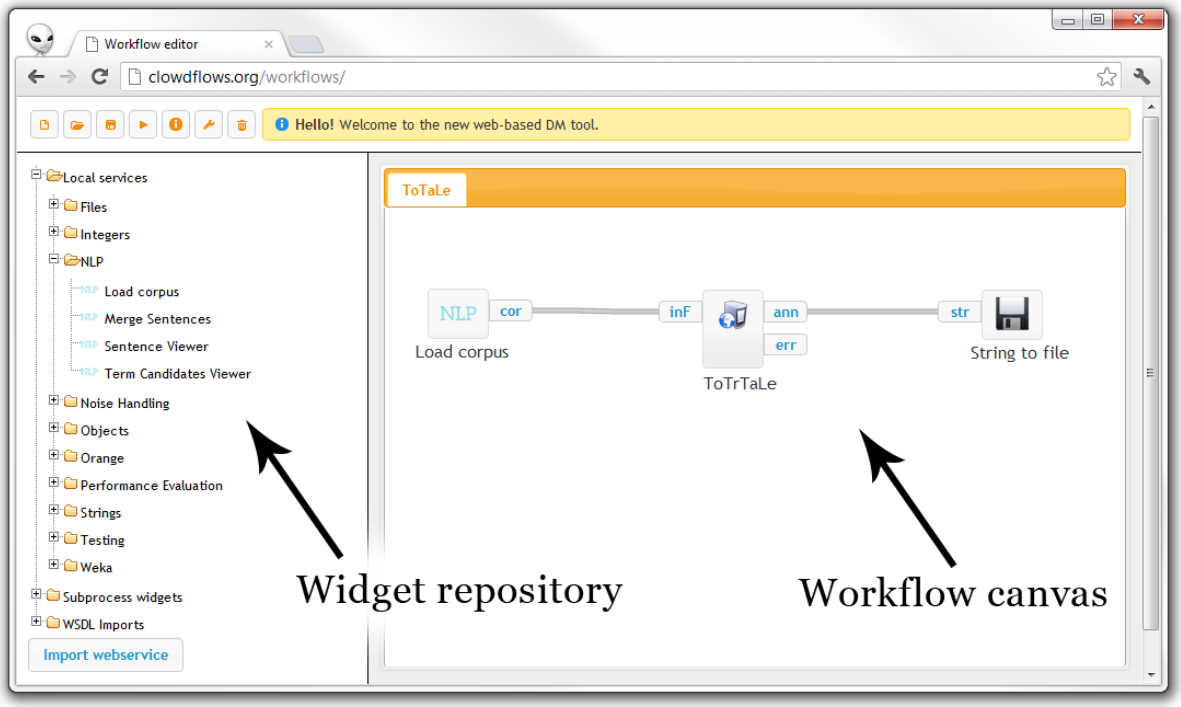

Figure 1. A screenshot of the ClowdFlows workflow editor in the Google Chrome browser and the ToTrTaLe workflow, available at http://clowdflows.org/workflow/228/.

Based on the file type, the program chooses the correct converter:

- If the document is an HTML document, its URL is written in the document variable and the document is assumed to contain only plain text. The web service then downloads the document via the given URL in plain text.
- DOCX Microsoft Word documents are essentially compressed ZIP files containing the parts of the document in XML.
- DOC Microsoft Word files are converted using an external tool, wvText (Lachowicz et al., 2006), which transforms the file into plain text.
- PDF files are converted with the Python pdfminer library (Shinyama, 2010).
- If the file name ends with TXT, then the file is assumed to be already in plain UTF-8 text.
- ZIP files are extracted into a flat directory and converted to a file with XML elements containing the plain-text of the individual files.

The resulting text representation is then sent through several regular expression filters, in order to further normalize the text. For instance, white space is normalised.

The final step involves sending the data. At each step of the web service process, errors are accumulated in the error output variable.

4.1.2. ToTrTaLe web service

The second web-service implements the ToTrTaLe annotation tool and also supports post-processing which corrects some systematic errors, which are further described in Section 5. The parameters of this web service are: the document, the language of the text (English, Slovene or historical Slovene), if we want post-processing, and if we want the output in XML format or as plain text.

The local ToTrTaLe service is then run, the output is written into the output variable, and the possible errors are passed to the error variable. Additionally, the input parameter for post-processing defines if the post–processing scripts are run on the text. The post-processing scripts are Perl implementations of corrections for tagging mistakes described in Section 5.

Finally, the output string variable is passed on to the output of the web service.

4.2. Implemented widgets

Apart from the web services we also needed to adapt some platform specific widgets to successfully use the web services. These widgets, not exposed as web services, are run locally; in the case of Orange4WS they are executed on the user's machine, whereas in the case of ClowdFlows they are executed on the server hosting the ClowdFlows application.

Orange4WS and ClowdFlows can automatically construct widgets for web services. They identify the inputs and the outputs of the web service from the service's WSDL description. Nevertheless, an additional functionality was required to adequately support the user in using the web services. Both in Orange4WS as well as in ClowdFlows, we implemented a widget called "Load Corpus" that opens a corpus in one of the formats supported by the web service for parsing input data, as well as internally calls the web service for converting input data.

4.3. Example workflow

The widgets implementing the existing software components are shown in Figure 1 and Figure 2. Figures show that the implementation of web-services is platform-independent.

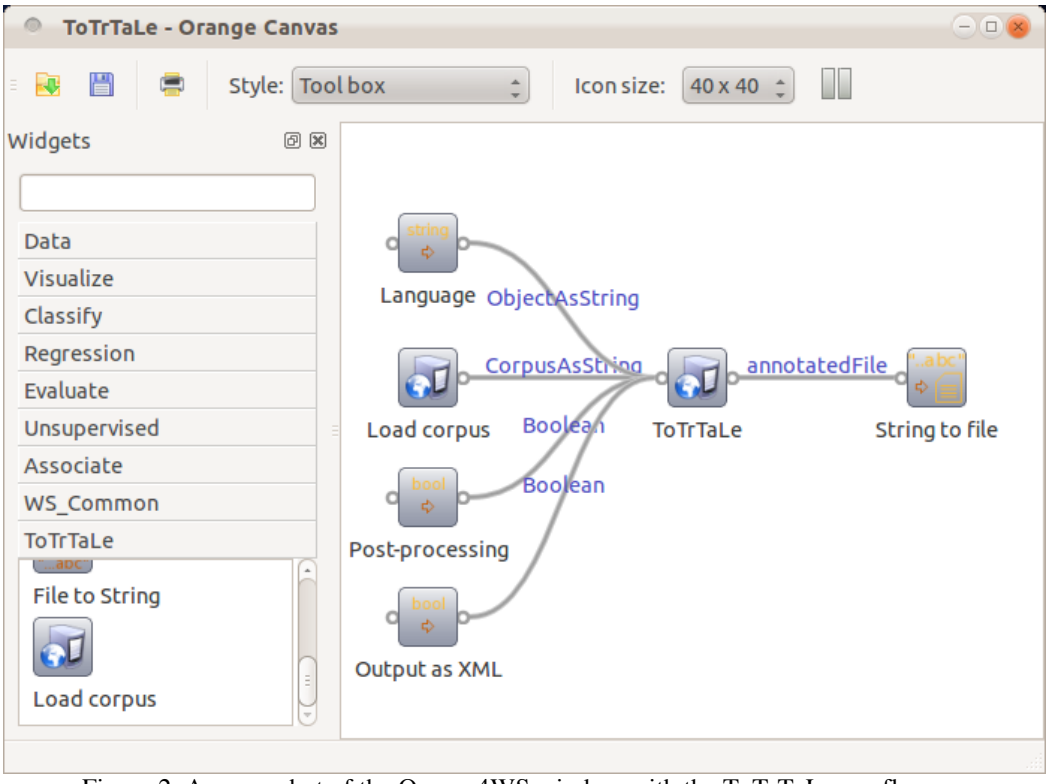

Figure 2. A screenshot of the Orange4WS window with the ToTrTaLe worflow.

In both figures the same workflow is represented. Figure 1 shows the workflow in the ClowdFlows platform and Figure 2 shows the workflow in the Orange4WS platform.

On the left side of both figures, there is a widget repository, and the right side is intended for the construction of workflows – the canvas. Apart from our web service widgets, there are some general-purpose widgets (e.g., file reading, file writing, construction of strings). The purpose of both workflows is essentially the same: they accept a file and read the file. Then the file is parsed from its original form into the plain text representation of the file. After the parsing of the file, the plain text representation is input into the ToTrTaLe web service. The service returns the annotated file in the plain text representation according to other input parameters. The final file can be viewed in the right most widget (String to file) of the corresponding workflow.

There is also a minor difference in the workflows presented in Figures 1 and 2. The difference is that the Orange4WS workflow has more widgets than the ClowdFlows workflow. This is due to the fact that widgets for Orange4WS were implemented to accept input data from other widgets (String widget, Boolean widget, etc.), whereas the widgets for ClowdFlows were implemented to accept inputs directly as parameters (by double clicking on the widget).

The sample output produced by either of the two workflows is shown in Figure 3. The figure clearly shows the function of each token, the sentence splitter tags and also morphosyntactic annotation of each token. The final output is in the form of plain text, where the input to the workflow was a Slovene PDF file.

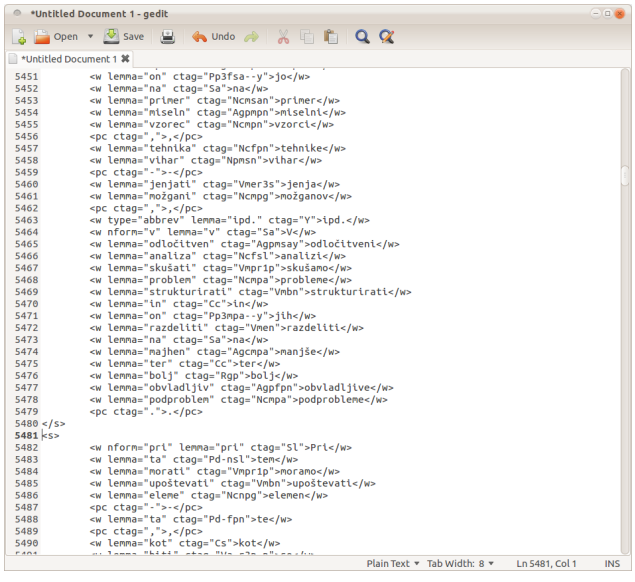

Figure 3. A sample output from the ToTrTaLe web-service, annotating sentences and tokens, with lemmas and MSD corpus tags on words.

5. Analysis of tagging mistakes

In this section we present the observed ToTrTaLe mistakes, mainly focusing on Slovene. The corpus used for analysis contains the papers of the Proceedings of the past seven Language Technology conferences. The construction of the corpus is described in Smailović and Pollak (2011).

The majority of the described mistakes are currently handled in an optional post-processing step, but can be taken into consideration in future versions of ToTrTaLe, by improving tokenisation rules or changing the tokeniser, re-training the tagger with larger and better corpora and lexica, and improving the lemmatisation models or learner.

5.1. Incorrect sentence segmentation

Errors in sentence segmentation originate mostly from the processing of abbreviations. Since the analysed examples were taken from academic texts, specific abbreviations, leading to incorrect separation of sentences, are frequent. The abbreviations that should be added to the abbreviation list for ToTrTaLe are e.g. "*et al.*", "*in sod.*", "*cca.*". On the other hand there are abbreviations after which ToTrTaLe should end the sentence, but doesn't. Checking if there is an upper case letter following the abbreviation would, in most cases, solve this mistake. Examples include the measures "*KB*", "*MB*", "*GB*", and "*ipd.*", "*itd.*", "*etc.*".

5.2. Incorrect morphosyntactic annotations

The tagging also at times makes mistakes, and in some cases these mistakes occur systematically. One example is in subject complement structures. For instance "*Kot podatkovne strukture so semantične mreže usmerjeni grafi.*" [*As data structures semantic networks are directed graphs.*] the nominative plural feminine "*semantične mreže*" [*semantic networks*] is wrongly annotated as singular genitive feminine. Another frequent type of mistake, easy to correct, is unrecognized gender/number/case agreement between adjective and noun in noun phrases. For example, "*Na eni strani imamo semantične leksikone …*" [*On the one hand we have the semantic lexicons…*], "*semantične*" [*semantic*] is assigned a feminine plural nominative MSD, while "*leksikone*" [*lexicons*] is attributed a masculine plural accusative tag. Next, in several examples, "*sta*" (second person, dual form of verb "*to be*") is tagged as a noun. Even if "*STA*" can be used as an abbreviation (when written with capital letters), it is much more frequent as the word-form of the auxiliary verb.

5.3. Incorrect lemmatisation

Besides the most common error of wrong lemmatisation of individual words (e.g. "*hipernimija*" being lemmatised as "*hipernimi*" [hypernyms] and not as "*hipernimija*" [*hypernymy*]), there are systematic errors when lemmatising Slovene adjectives in comparative and superlative form, where the base form is not chosen as a lemma. Last but not least, there are typographic mistakes in the original text and of end-of-line split words.

6. Conclusions and further work

In this paper we presented the ToTrTaLe web service and demonstrated how it can be used in workflows in two service-oriented data mining platforms – Orange4WS and ClowdFlows. Together with the ToTrTaLe web service, we developed a series of widgets (workflow components) for pre-processing the text, consisting of reading the text corpus files in various formats, tokenising the text, lemmatising and morphosyntactically annotating it, as well as adding the sentence boundaries, followed by a post-processing widget for error correction.

Before starting this work, the To(Tr)TaLe tool has already existed as a web application for Slovene, where the user was able to upload and add the text, but the novelty is that a web service implementation now enables the user to use ToTrTaLe as a part for various other NLP applications. The presented web service has already been incorporated in the term and definition extraction workflow[2] (Pollak et al. 2012).

Acknowledgements

We are grateful to Vid Podpečan and Janez Kranjc for their support and for enabling us to include the developed widgets into Orange4WS and ClowdFlows, respectively. This work was partially supported by the Slovene Research Agency and the FP7 European Commission projects "Machine understanding for interactive storytelling" (MUSE, grant agreement no: 296703) and "Large scale information extraction and integration infrastructure for supporting financial decision making" (FIRST, grant agreement 257928).

References

Špela Arhar Holdt and Vojko Gorjanc (2007). Korpus FidaPLUS: nova generacija slovenskega referenčnega korpusa. *Jezik in slovstvo*, 52(2): 95–110.

Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel and Bernd Wiswedel (2007). KNIME: The Konstanz Information Miner. In Preisach, C., Burkhardt, H., Schmidt-Thieme, L., Decker, R., (eds.): *GfKl. Studies in Classification, Data Analysis, and Knowledge Organization*, Springer, pp. 319–326.

Thorsten Brants (2000). TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of the 6h Applied Natural Language Processing Conference* (ANLP 2000), Seattle, WA, pp. 224–231.

Janez Demšar, Blaž Zupan, Gregor Leban and Tomaž Curk (2004). Orange: From experimental machine learning to interactive data mining. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.): *Proceedings of ECML/PKDD-2004*. Springer LNCS Volume 3202, pp. 537–539.

Tomaž Erjavec (2011). Automatic linguistic annotation of historical language: ToTrTaLe and XIX century Slovene. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, ACL.

Tomaž Erjavec (2012a). The goo300k corpus of historical Slovene. In *Proceedings of the 8th conference on Language Resources and Evaluation (LREC'12)*, Istambul. European Language Resources Association (ELRA).

Tomaž Erjavec (2012b). MULTEXT-East: morphosyntactic resources for Central and Eastern European languages. *Language resources and evaluation*, 46(1): 131–142.

Tomaž Erjavec and Sašo Džeroski (2004). Machine Learning of Language Structure: Lemmatising Unknown Slovene Words. *Applied Artificial Intelligence*, 18(1):17–41.

Tomaž Erjavec, Darja Fišer, Simon Krek and Nina Ledinek (2010). The JOS linguistically tagged corpus of Slovene. In *Proceedings of the 7th International Conference on Language Resources and Evaluations*, LREC 2010, Valletta, Malta, pp. 1806-1809.

Tomaž Erjavec, Camelia Ignat, Bruno Pouliquen and Ralf Steinberger (2005). Massive Multi-Lingual Corpus Compilation: Acquis Communautaire and ToTaLe. In *Proceedings of the 2nd Language & Technology Conference*, April 21-23, 2005, Poznan, Poland, pp. 32–36.

Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Matthew R. Pocock, Peter Li and Thomas M. Oinn (2006). Taverna: A tool for building and running workflows of services. *Nucleic Acids Research 34* (Web-Server-Issue): 729–732.

Dom Lachowicz and Caolán McNamara (2006). *wvWare, library for converting Word document*. http://wvware.sourceforge.net/, accessed in August 2012.

Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz and Timm Euler (2006). YALE: rapid prototyping for complex data mining tasks. In Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D. (eds.): *Proceedings of KDD-2006*, ACM , pp. 935–940.

Janez Kranjc, Vid Podpečan, and Nada Lavrač (2012). ClowdFlows: A cloud-based scientific workflow platform. In *Proceedings of ECML/PKDD-2012*. September 24-28, 2012, Bristol, UK, *Springer LNCS* (in press).

Vid Podpečan, Monika Žakova and Nada Lavrač (2012). Orange4ws environment for service-oriented data mining. *The Computer Journal* (2012), 55(1): 82–98.

Senja Pollak, Anže Vavpetič, Janez Kranjc, Nada Lavrač and Špela Vintar (2012). In J. Jancsary (ed.): *Proceedings of the 11th Conference on Natural Language Processing* (KONVENS 2012), September 19-21, 2012, Vienna, Austria, pp. 53–60.

Yusuke Shinyama (2010). PDFMiner http://www.unixuser.org/~euske/python/pdfminer/index.html, accessed in August 2012.

Jasmina Smailović and Senja Pollak (2011). Semi-automated construction of a topic ontology from research papers in the domain of language technologies. In *Proceedings of the 5th Language & Technology Conference*, November 25–27, 2011, Poznan, Poland, pp. 121–125.

TEI Consortium (2007). *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. http://www.tei-c.org/Guidelines/P5/

Ian H. Witten, Eibe Frank and Mark Hall (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Third Edition. Morgan Kaufmann.

[2] http://clowdflows.org/workflow/76/