NLP Web Services for Slovene and English: Morphosyntactic Tagging, Lemmatisation and Definition Extraction

Senja Pollak, Nejc Trdin, Anže Vavpetič and Tomaž Erjavec
Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia
E-mail: {senja.pollak, nejc.trdin, anze.vavpetic, tomaz.erjavec}@ijs.si

Keywords: web services, workflows, morphosyntactic tagging, lemmatisation, definition extraction

Received: November 30, 2012

This paper presents a web service for automatic linguistic annotation of Slovene and English texts. The web service enables text up-loading in a number of different input formats, and then converts, tokenises, tags and lemmatises the text, and returns the annotated text. The paper presents the ToTrTaLe annotation tool, and the implementation of the annotation workflow in two workflow construction environments, Orange4WS and ClowdFlows. It also proposes several improvements to the annotation tool based on the identification of various types of errors of the existing ToTrTaLe tool, and implements these improvements as a post-processing step in the workflow. The workflows enable the users to incorporate the annotation service as an elementary constituent for other natural language processing workflows, as demonstrated by the definition extraction use case.

Povzetek: Prispevek predstavi spletni servis ToTrTaLe za jezikoslovno označevanje slovenskega in angleškega jezika, njegovo implementacijo v okoljih za gradnjo delotokov Orange4WS in ClowdFlows ter njegovo uporabo v delotoku za luščenje definicij.

1 Introduction

In natural language processing (NLP), the first steps to be performed on the input text are tokenisation, part-of-speech tagging and lemmatisation. The output of these three steps is a string of text tokens, where each word token is annotated with its context disambiguated part-of-speech tag and the base form of the word, i.e. lemma, thus abstracting away from the variability of word-forms. For example, the Slovene sentence "*Hotel je dober hotel*" ("[*He*] *wanted a good hotel*") can be lemmatised and tagged as "*hoteti/Verb biti/Verb dober/Adjective hotel/Noun*"; as can be seen, the first and last word tokens are the same, yet their part of speech and lemma differ.

Such annotation is very useful for further processing, such as syntactic parsing, information extraction, machine translation or text-to-speech, to mention just a few. However, all three processing steps (tokenisation, part-of-speech tagging and lemmatisation) are language dependent, and software to perform them is—especially for smaller languages—often not available or difficult to install and use.

Recently, there has been an upsurge of interest in workflow construction environments, the best known being Taverna (Hull et al., 2006) developed for workflow composition and execution in the area of bioinformatics. In such workflow environments it is not necessary to locally install a tool used as a workflow ingredient, but rather use web services available elsewhere, and link them together into workflows. This frees the users from installing the needed tools (which might not be available for downloading in any case) and, indeed, from needing high-end computers to perform computationally demanding processing over large amounts of data. While online workflow construction tools are already widely used in some domains, this approach has only recently started being used also in the field of NLP (Pollak et al., 2012a).

This paper, extending our previous work on this topic (Pollak et al., 2012b), focuses on a particular tool for automatic morphosyntactic tagging and lemmatisation, named ToTrTaLe (Erjavec, 2011), currently covering two languages, Slovene and English. Its description 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 and workflows and by presenting two specific workflow construction environments, Orange4WS (Podpečan et al., 2012) and ClowdFlows (Kranjc et al., 2012). 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 the two workflow construction environments, while Section 5 presents some improvements of the ToTrTaLe tool based on the identification of several types of errors of the existing implementation. The utility of the ToTrTaLe web service as a pre-processing step for other NLP tasks is illustrated by a definition extraction use case in Section 6. Finally, Section 7 gives conclusions and directions for further work.

2 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 largely language independent and relies on external modules to perform the specific language processing tasks. 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[1]. It allows pasting the input text into the form or uploading it as a plain-text UTF-8 file, while the annotated output text can be either 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). An additional extension of ToTrTaLe is the ability to process heavily annotated XML document conformant to the Text Encoding Initiative Guidelines (TEI, 2007).

The rest of this section presents the main modules of ToTrTaLe and their models for Slovene and English, leaving out the description of the historical language models which are out of the main scope of this paper.

2.1 Tokenisation

The multilingual tokenisation module mlToken[2] is written in Perl and in addition to splitting the input string into tokens also assigns to each token its 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. Furthermore, the tokeniser also segments the input text into sentences.

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 for both languages.

2.2 Tagging

Part-of-speech tagging is the process of assigning a word-level grammatical tag to each word in running text, where the tagging is typically performed in two steps: the lexicon gives the possible tags for each word, while the disambiguation module assigns the correct tag based on the context of the word.

Most contemporary taggers are trained on manually annotated corpora, and the tagger we use, TnT (Brants, 2000), is no exception. TnT is a fast and robust tri-gram tagger, which is also able, by the use of heuristics over the words in the training set, to tag unknown words with reasonable accuracy.

For languages with rich inflection, such as Slovene, it is better to speak of morphosyntactic descriptions (MSDs) rather than part-of-speech tags, as MSDs contain much more information than just the part-of-speech. For example, the tagsets for English have typically 20–50 different tags, while Slovene has over 1,000 MSDs.

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 we use 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. The lemmatisation of 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] The application is available at http://nl.ijs.si/jos/analyse/

[2] 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". Web service 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.

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 latter being a web environment, which is not the case for the first one.

3.1 The Orange4WS platform

The first platform, Orange4WS (Podpečan et al., 2012), is a data mining platform distinguished 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.

Orange4WS is built on top of two open source projects: (a) the Orange data mining framework (Demšar et al., 2004), which provides the Orange canvas for constructing workflows as well as core data structures and machine learning algorithms, and (b) the Python Web Services project[3] (more specifically, the Zolera SOAP infrastructure), which provides the libraries for developing web services in the Python programming language.

Furthermore, in contrast with other workflow environments Orange4WS offers a rather unique combination of features, mainly:

- A large collection of data mining and machine learning algorithms,
- A collection of powerful yet easy to use visualization widgets and
- Easy extendibility either in Python or C++ due to layered architecture of the Orange environment.

Unlike ClowdFlows (as will be explained in the next section) the user is required to install Orange4WS on her own machine in order to create and execute workflows. Furthermore, local widgets (widgets that are not implemented as web services) are executed on the client's computer, thus using its computational resources, which can quickly become a problem when solving more complex tasks.

3.2 The ClowdFlows platform

The second platform ClowdFlows (Kranjc et al., 2012) is distinguished from other main data mining platforms especially by the fact that it requires no installation from the user and can be run on any device with an internet connection, using any modern web browser. Furthermore, ClowdFlows also natively supports workflow sharing between users.

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.

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. ClowdFlows consists of the browser-based workflow editor and the server-side application which handles the execution of 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

[3] http://pywebsvcs.sourceforge.net/

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.

4 Implementation of the ToTrTaLe Web Service and Workflows

In this section we present two web services that we implemented and also some details regarding the implementations. The services were implemented in the Python programming language, using Orange4WS API and additional freeware software packages used for enabling different input types. Services are currently adapted to run on Unix-like operation systems, but are easily transferable to other operation systems. In addition, the workflows constructed using these web services are also presented.

4.1 Implemented web service

The implemented web service constitutes the main implementation part of this work. The web service has two functionalities: the first converts different input files to plain text format, while the second uses the ToTrTaLe tool to annotate input texts. The two functionalities correspond to two operations described in one WSDL file. In this section we give the descriptions of both functionalities, together with some implementation details.

4.1.1 Converting input files to plain text

The first operation of the web service parses the input files and converts them into plain text. The input corpus file can be uploaded in various formats, either as a single file or as several files compressed in a single ZIP file. The supported formats are PDF, DOC, DOCX, TXT and HTML, the latter being passed to the service in the form of an URL as a document. Before being transferred, the actual files are encoded in the Base64 representation, since some files might be binary files. So the first step is to decode the Base64 representation of the document.

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

- If the file extension is HTML, we assume that an URL address is passed and that it is written in the document variable. It is also assumed that the document contains 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. The content of the file is first unzipped, and then all the plain text is extracted.
- DOC Microsoft Word files are converted using an external tool, wvText (Lachowicz and McNamara, 2006), which transforms the file into plain text. The tool is needed because the whole file is a compiled binary file and it is hard to manually extract the contents without appropriate tools.
- PDF files are converted with the Python pdfminer library (Shinyama, 2010). The library is a very good implementation for reading PDF files, with which one can extract the text, images, tables, etc., from a PDF file.
- If the file name ends with TXT, then the file is assumed to be already in plain UTF-8 text format. The file is only read and sent to the output.
- ZIP files are extracted into a flat directory and converted appropriately—as above—based on the file extension. Note that ZIP files inside ZIP files are not permitted.

The resulting text representation is then sent through several regular expression filters, in order to further normalize the text. For instance, white space characters are merged into one character.

The final step involves sending the data. But before that, the files have their unique identifiers added to the beginning of the single plain text file. The following steps leave these identifiers untouched, so the analysis can be traced through the whole workflow. At each step of the web service process, errors are accumulated in the error output variable.

4.1.2 Tokenisation, tagging and lemmatisation

The second operation of the web service exposes the ToTrTaLe annotation tool. The mandatory parameters of this operation are: the document in plain text format and the language of the text (English, Slovene or historical Slovene). Non-mandatory parameters are used to determine whether the user wants post-processing (default is no), and whether the output should be in the XML format (default) or in the plain text format.

Both Orange4WS and ClowdFlows send the data and the processing request to the main web service operation, i.e. ToTrTaLe annotation, which is run on a remote server. 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 and the accumulated errors are passed on to the output of the web service, which is then sent back to the client.

4.1.3 Implemented widgets

Orange4WS and ClowdFlows can automatically construct widgets for web services, where each operation maps into one widget (thus, the web service described in this paper maps into two widgets). They identify the inputs and the outputs of the web service's operations from the WSDL description. In addition to implementing the web service operations described above, additional functionality was required to adequately support the user in using this web service and some additional platform specific widgets were implemented accordingly. 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.

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, and internally calls the service's operation for converting input data. They essentially read the user selected files, encode them in Base64 and send the file to the web service. Widgets return the output produced by the web service.

4.2 ToTrTaLe workflows

The widgets implementing the existing software components are incorporated into the workflows presented in Figure 1 and Figure 2. The figures show that the implementation of the web service is platform-independent. In both figures the same workflow is shown: Figure 1 shows the workflow in the Orange4WS platform and Figure 2 the workflow in the ClowdFlows platform. On the left side of both figures, there is a widget repository, and the right side presents the canvas used for workflow construction. Apart from our web service widgets, the workflows contain also 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 by the "Load corpus" widget. After the parsing of the file, the plain text representation is input into the ToTrTaLe widget. The widget returns the annotated file in plain text or XML representation according to one of the input parameters. The final file can be viewed in the rightmost widget (String to file) of the corresponding workflows.

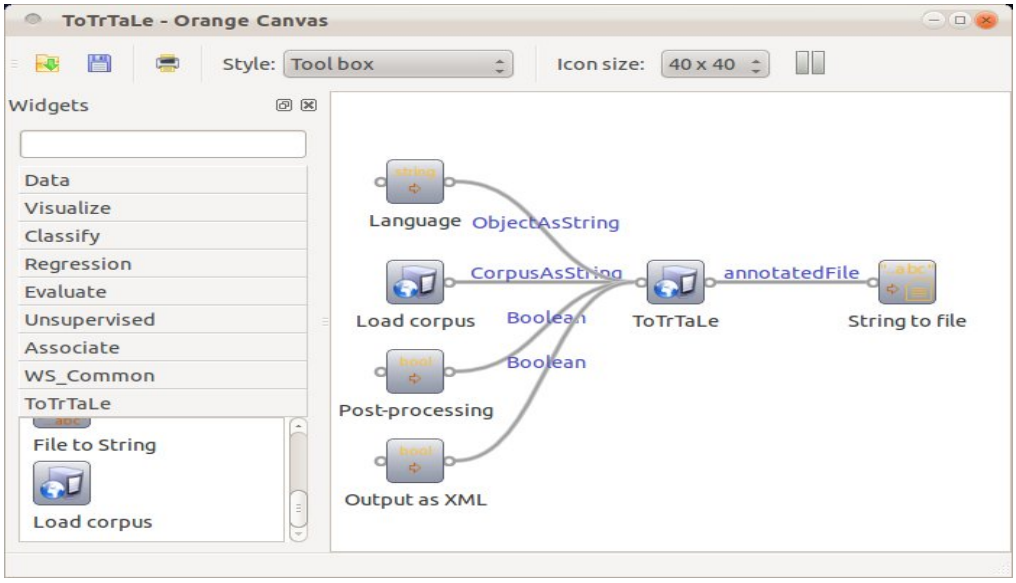

Figure 1: A screenshot of the ToTrTaLe workflow in the Orange4WS workflow editor.

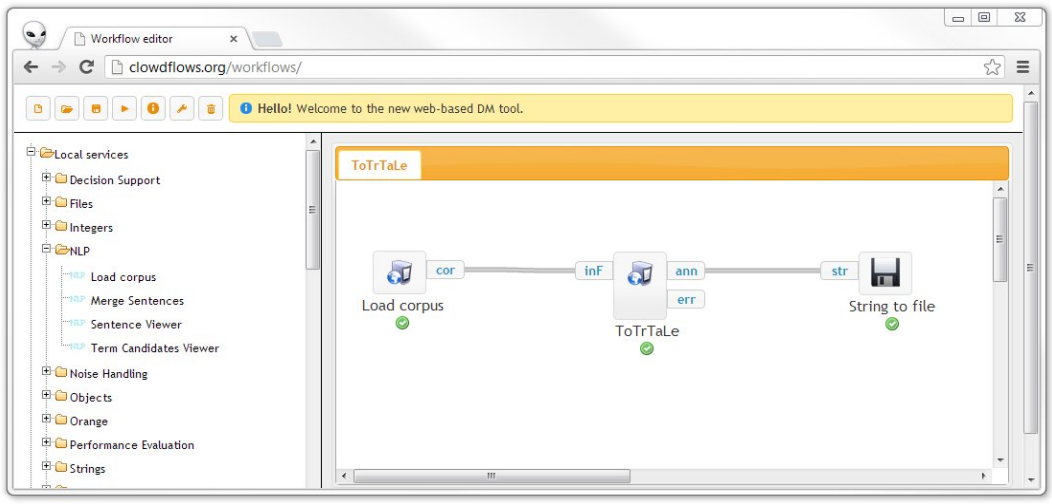

Figure 2: A screenshot of the the ToTrTaLe workflow in the ClowdFlows workflow editor, available online at http://clowdflows.org/workflow/228/.

There is also a minor difference in the workflows presented in Figures 1 and 2: 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 the 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.

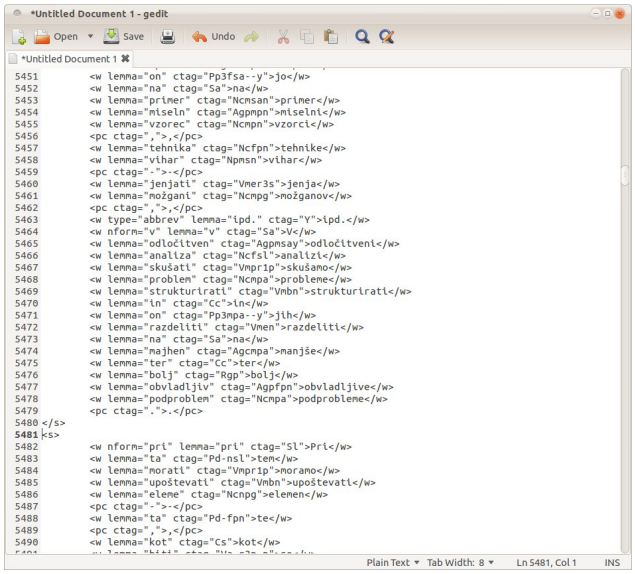

Figure 3: A sample output from the ToTrTaLe web service, annotating sentences and tokens, with lemmas and MSD tags on words.

5 Improving ToTrTaLe Through Post-processing based on the Analysis of Annotation Mistakes

In this section we present the observed ToTrTaLe mistakes, focusing on Slovene, and propose some corrections to be performed in the post-processing step. The corpus used for the analysis consists of the papers of seven consecutive proceedings of Language Technology conferences, held between 1998 and 2010. The construction of the corpus is described in Smailović and Pollak (2011).

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.

In some examples the abbreviations contain the period that is—if the abbreviation is not listed in the abbreviation repository—automatically interpreted as the end of the sentence. For instance, abbreviation “et al.”, frequently used in referring to other authors in academic writing therefore incorrectly implies the end of the sentence, and the year of the publication is mistakenly treated by ToTrTaLe as the start of a new sentence. This is now corrected in ToTrTaLe post-processing.

Note, however, that the period after the abbreviation does not always mean that the sentence actually continues. This is the case when an abbreviation occurs at the end of the sentence (“ipd.”, “itd.”, “etc.” are often in this position). Consequently, in some cases two sentences are mistakenly tagged by ToTrTaLe as a single sentence. This mistake was also observed with the abbreviations EU or measures KB, MB, GB if occurring at the last position of the sentence just before the period.

5.2 Incorrect morphosyntactic annotations

The tagging also at times makes mistakes, some of which occur systematically. One example is in subject complement structures. For instance, in the Slovene sentence “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, in the sentence “Na eni strani imamo semantične leksikone …” [On the one hand we have 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 due to end-of-line split words.

5.4 ToTrTaLe post-processing

The majority of the described mistakes are currently handled in an optional post-processing step, but should 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.

In the current post-processing implementation we added a list of previously unrecognized abbreviations (such as "et al.", "in sod.", "cca.") to avoid incorrect redundant splitting of the sentence.

We corrected the wrongly merged sentences by splitting them into two different sentences if certain abbreviations (such as "etc.") are followed by an upper-case letter in the word following the abbreviation.

Other post-processing corrections include the correction of adjective-noun agreement, where we assume that the noun has the correct tag and the preceding adjective takes its properties.

Some other individual mistakes are treated in the post-processing script, but not all the mistakes have been addressed.

6 Use Case: Using ToTrTaLe in the Definition Extraction Workflow

In this section we present the usefulness of the presented annotation web service implementation for the task of definition extraction.

The definition extraction workflow, presented in detail in Pollak et al. (2012a), was implemented in the ClowdFlows platform and includes several widgets. The workflow starts with two widgets presented in the previous sections:

- Load corpus widget, which allows the user to conveniently upload her corpus in various formats, and
- ToTrTaLe tokenization, morphosyntactic annotation and lemmatization service for Slovene and English.

The workflow's main components for definition extraction are implemented in the following widgets:

- *Pattern-based definition extractor*, which seeks for sentences corresponding to predefined lexico-syntactic patterns (e.g., NP [nominative] is a NP [nominative]),
- *Term recognition-based definition extractor*, which extracts sentences containing at least two domain-specific terms identified through automatic term recognition,
- *WordNet- and sloWNet-based definition extractor*, which identifies sentences containing a wordnet term and its hypernym.

In addition, several other widgets have been implemented (Pollak et al., 2012a):

- Term extractor widget implementing the LUIZ term recognition tool (Vintar, 2010) that we can use separately for extracting the terms from the corpus as well as the necessary step for the second definition extraction method,
- Term candidate viewer widget, which formats and displays the terms (and their scores) returned by the term extractor widget,
- Sentence merger widget, which allows the user to join (through intersection or union) the results of several definition extraction methods,
- Definition candidate viewer widget, which, similarly to the term candidate viewer widget, formats and displays the candidate definition sentences returned by the corresponding methods.

The three definition extraction methods, implemented as separate operations of one web service, are described in some more detail below.

- The first approach, implemented in the *pattern-based definition extraction* widget, is the traditional pattern-based approach. We created more than ten patterns for Slovene, using the lemmas, part-of-speech information as well as more detailed morphosyntactic descriptions, such as case information for nouns, person and tense information for verbs, etc. The basic pattern is for instance "NP-nom Va-r3[psd]-n NP-nom" where "NP-nom" denotes a noun phrase in the nominative case and the "Va-r3[psd]-n" matches the auxiliary verb in the present tense of the third person singular, dual or plural and the form is not negative, in other words it corresponds to "je/sta/so" [is/are] forms of the verb "biti" [to be]. As there is no chunker available for Slovene, the basic part-of-speech annotation provided by ToTrTaLe was needed for determining the possible noun phrase structures and the positions of their head nouns.
- The second approach, implemented in the *term recognition-based definition extraction* widget, is primarily tailored to extract knowledge-rich contexts as it focuses on sentences that contain at least *n* domain-specific single or multi-word terminological expressions (terms). The parameters of this module are the number of terms, the number of terms in the nominative case, if a verb should figure between two terms, if the first term should be a multi-word term and if the sentence should begin with a term. For setting these parameters, the ToTrTaLe information was needed.
- The third approach, implemented in the *WordNet-based definition extraction* widget, seeks for sentences where a wordnet term occurs together with its direct hypernym. For English we use the Princeton WordNet (PWN) (Fellbaum, 1998), whereas for Slovene we use sloWNet (Fišer and Sagot, 2008), a Slovene counterpart of WordNet.

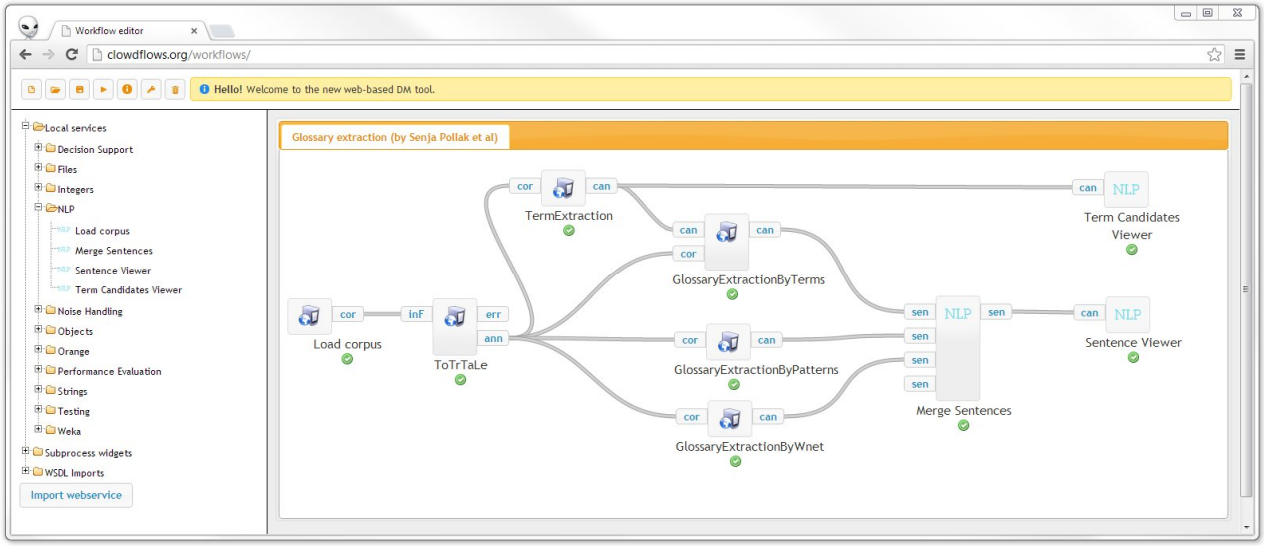

Figure 4: The definition extraction workflow (http://clowdflows.org/workflow/76/).

7 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, initially presented in Pollak et al. (2012b), the ToTrTaLe 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. For illustration, this paper presents the use case of ToTrTaLe in an elaborate workflow, which implements definition extraction for Slovene and English.

In further work we plan to develop other workflows for the processing of the natural language, especially for Slovene, where the ToTrTaLe web service will be used as the initial step.

Acknowledgement

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. The definition extraction methodology was done in collaboration with Špela Vintar and Darja Fišer. 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

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

[2] 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, 319–326.

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

[4] 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, 537–539.

[5] 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.

[6] Tomaž Erjavec (2012a). The goo300k corpus of historical Slovene. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, LREC 2012, Istanbul, Turkey, 2257–2260.

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

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

[9] 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, 1806–1809.

[10] 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, 32–36.

[11] Christiane Fellbaum (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press. Online version: http://wordnet.princeton.edu.

[12] Darja Fišer and Benoît Sagot (2008). Combining multiple resources to build reliable wordnets. *Text, Speech and Dialogue* (LNCS 2546). Berlin; Heidelberg: Springer, 61–68.

[13] 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.

[14] 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, 816–819.

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

[16] 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 , 935–940.

[17] Vid Podpečan, Monika Žakova and Nada Lavrač (2012). Orange4WS environment for service-oriented data mining. *The Computer Journal* (2012) 55(1): 82–98.

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

[19] Senja Pollak, Nejc Trdin, Anže Vavpetič and Tomaž Erjavec (2012b). A Web Service Implementation of Linguistic Annotation for Slovene and English. In *Proceedings of the 8th Language Technologies Conference, Proceedings of the 15th International Multiconference Information Society* (IS 2012), Volume C, 157–162.

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

[21] 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, 121–125.

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

[23] Špela Vintar (2010). Bilingual term recognition revisited: The bag-of-equivalents term alignment approach and its evaluation. *Terminology* 16(2): 141–158.

[24] Ian H. Witten, Eibe Frank and Mark Hall (2011). *Data Mining: Practical Machine Learning Tools and Techniques.* 3rd Edition. Morgan Kaufmann.