Language Technologies and Corpus Encoding

Tomaž Erjavec, Peter Holozan and Nikola Ljubešić

Abstract

This chapter provides an overview of the levels of basic automatic linguistic annotation that should be applied to corpora to be used as the basis for lexicographic analyses of contemporary Slovene, as well as for other purposes. We give an overview of existing research in this field and then focus on a concrete set of open source and mainly language independent tools and their models for Slovene, and give suggestions for their improvement. A short description of the proposed corpus encoding process is also provided.

Keywords: linguistic annotation, corpora, annotation format

1 INTRODUCTION

This chapter deals with linguistic annotation of corpora which could serve as the basis for lexicographic work, and also provides suggestions for the format of the corpus annotations. The chapter does not cover all tools that are useful for lexicographic work, but only those that generate annotations to be included in the corpus and then used as a source of knowledge by down-stream programs, from concordancers to synonym extractors. In addition, the focus is predominantly on programs that have been developed for the Slovene language. The following levels of annotation will be taken into account, and are listed in typical order of appearance in the processing chain:

1. **Tokenisation**, which divides the text into individual tokens, either words or punctuation. This step can also identify token types, such as numerals, abbreviations, URLs, emoticons and emoji. Sentence segmentation is often performed in the same step.
2. **Normalisation**, which transforms (translates) non-standard word forms (found, for example, in historical texts and in user-generated content) into standard ones. This is useful for easier searching and better performance of the annotation tools developed for standard language.
3. **Morphosyntactic (or part-of-speech) tagging**, which assigns to each word token a morphosyntactic description (MDS), e.g. *Ncmdn* which decomposes to a common noun of masculine gender, dual number and nominative case.
4. **Lemmatisation**, which assigns the base form to a word, used for dictionaries or lexical look-up.
5. **Parsing**, which gives a syntactic analysis to each sentence of the text.

Apart from parsing, all the above levels of annotation are necessary for a corpus to be useful in lexicographic work. Some other levels of annotation might also be useful, but these are difficult to place at a fixed position in the processing chain, as this depends if they require (or can use) all or some of the above annotations. Depending on the method, these further annotation tools can use raw or tokenised text, which could furthermore be tagged or even parsed. Some of these tools have already been developed for Slovene, even if only as prototypes:

6. **Named entity recognition**, which identifies proper names in the text and classifies them, for example, into personal names, geographical names, and names of companies or institutions. Additionally, some systems identify numerical and other expressions and classify them, such as into currencies, dates and so on.

7. **Term extraction**, which identifies potential terms in the text. However, it should be noted that what constitutes a term is fairly problematic, as this depends on the subject area, target audience and the like.
8. **Semantic information annotation**, which labels words or phrases with their meaning by relying on a semantic and lexical resource; it can also link them together according to their semantic roles. Although such annotations could be extremely useful for lexicographers, the complexity of the annotation causes existing programs to generate results that may not be accurate enough to be of value.

Generally speaking, annotation (and other) language technology tools come in two varieties:

- Tools that use handwritten rules, which require a lot of human work but may give (depending on the level of annotation) very good results. Such tools are often used for text segmentation, e.g. into tokens or terms, and traditionally also for morphological analysis. For some levels of annotation, most notably morphosyntax and syntax, the number of necessary rules becomes extremely large, which makes their development and debugging very difficult, costly and error prone.
- Tools that learn a language model from training data, i.e. manually annotated corpora or other language resources. Machine learning methods are being developed at a fast pace, yet in order to generate quality models we typically need extensive language resources – and building these is a time-consuming and expensive process. On the other hand, once a training dataset has been built, it can be used to train and test various machine learning tools with the best one chose for the task at hand.

Both types of tools generally use background language resources, especially lexicons.

2 OVERVIEW OF TOOLS FOR THE SLOVENE LANGUAGE

Tools for annotating Slovene language texts at all the above-listed levels have already been developed, although a number remain at the prototype stage. This section will only focus on those which are still being maintained and, for the most part, which are freely or openly available. As such, one of the earliest tools for morphosyntactic annotation of Slovene (Jakopin and Bizjak Končar 1997) will not be included.

2.1 Tools developed by Amebis

The tools developed by Amebis are not openly accessible, yet their system of annotation tools and background resources – not only adapted to Slovene but written exclusively for it – has the longest tradition of development. These tools were initially developed for the Besana grammar checker (Holozan 2012) and Presis machine translator (Romih and Holozan 2002). They are written in the C++ programming language and work in 32- as well as 64-bit versions. Their structure slightly differs from the classic one: tokenisation is still performed first, but its key feature is the treatment of special tokens such as Web and e-mail addresses, phone numbers and emoticons as one unit.

The next step is the tagger, which uses a lexicon to annotate words with all possible combinations of lemmas and morphosyntactic tags (there are currently 7.6 million elements in the lexicon). The tagger also recognizes special tokens such as Web addresses, chemical formulas and emoticons. Simultaneously, it searches for potentially misspelled words and typical non-standard forms; some of the latter are already included in the lexicon with special morphosyntactic tags. The last step is performed by the analyser, which chooses the most likely pair of the lemma and morphosyntactic tag for each word. At the same time, it performs syntactic analysis and lists word meanings taken from the Ases database (Arhar and Holozan 2009), both by using an interface language developed at Amebis (Holozan 2011). If necessary, the analyser can also modify tokenisation or textual segmentation, e.g. when tackling examples such as "*ga*" as the pronoun "*him*" in Slovene, or "*ga.*" as an abbreviation equivalent to "*Mrs.*" in English. Examples such as "*Prišla je še ga. Micka.*" (*Mrs. Mary also came.*) are easy to handle, but cases such as "*Videl sem ga. Micka ga je tudi videla.*" (*I saw him. Mary saw him also.*), are much harder and may cause a naïve tokenizer to fail and identify the first instance of "*ga.*" as one token (abbreviation), and the text as one sentence, while the Besana tool would correctly identify two tokens (pronoun followed by a comma) and two sentences.

Amebis' tools rely on handwritten rules and data from the Ases database for their operation. The most important concept from the database is verb templates (Holozan 2011), which comprise data on valency. Many proper names have also been entered; they are divided into 20 categories (which enables named entity recognition). A special script tokenises, lemmatises and assigns MSD tags according to the specifications of the "Communication in Slovene" project and implemented in the Obeliks tagger (more on this in one of the following sections).

The tools by Amebis were also used to annotate two large reference corpora, Fida and FidaPLUS. The main obstacle to the wider usability of Amebis' tools

is license ownership. The tools are currently not open source, and thus an agreement needs to be signed with Amebis for their use.

2.2 The To(Tr)TaLe tagger

The ToTaLe tagger (Erjavec et al. 2005) was developed at the Department of Knowledge Technologies at the Jožef Stefan Institute within the framework of several projects. The tool implements a pipeline composed of three modules: a tokenizer, which also segments the text into sentences, a MSD tagger and a lemmatiser. A module called mlToken is used for tokenisation; it is a multilingual tokenizer that uses language dependent lists, e.g. of abbreviations or rules on how to write numbers, to adapt to a particular language. MSD tagging is done with the TnT tagger (Brants 2000), a relatively old trigram-based tagger which uses models trained for a specific language on a manually annotated corpus; it can also use a background lexicon. The current model is trained on the jos1M corpus (Erjavec et al. 2010; Erjavec and Krek 2010), and uses tokens from the FidaPLUS corpus (Arhar and Gorjanc 2007) as a background lexicon. The lemmatisation module uses a program called CLOG (Erjavec and Džeroski 2004), which assigns the base form to each word form according to its MSD tag. This program also relies on an automatic lemmatisation model, based on a training dataset, which consists of a list of triplets (word form, MSD tag, lemma). The training set for Slovene has been generated by combining the tokens from jos100k, manually checked tokens from jos1M and selected words from FidaPLUS. ToTaLe is available online and has been used to annotate most corpora that can be accessed through the noSketchEngine (Rychly 2007) concordancer installed at nl.ijs.si (Erjavec 2013).

ToTaLe is written in the Perl programming language, and the same applies to modules for tokenisation and lemmatisation. Although Perl is no longer a very popular language, it can still be used with all main operating systems. However, the TnT tagger is not open source, and is only available for non-commercial use as an executable under Linux, which is why in its current state ToTaLe cannot be made openly accessible nor used on OS Windows.

A tool called ToTrTaLe has also been developed, and this differs improves on ToTaLe in two important ways. First it includes an (optional) transcription module and, second, unlike ToTaLe, which expects raw text as input and outputs a tabular file, ToTrTaLe expects a TEI-compliant XML file at input and also returns a TEI XML file as output. The transcription module is intended for modernising historical word forms in older (Slovene) texts; by working on normalised forms, the MSD tagger and lemmatiser produce much better results, as they are both trained to process texts in contemporary Slovene. For

modernising the tokens, the transcription module uses a tool called Vaam (Reffle 2011), which uses handwritten rules on how to modernise historical Slovene word forms. To date, only the IMP corpus of historical Slovene (Erjavec 2015) has been annotated with ToTrTaLe.

Both tools give relatively good results, but their maintenance could be improved. For example, it would be worth re-training the models for the Slovene language, since better resources have since appeared, most notably the Sloleks lexicon (Arhar 2009; Dobrovoljc et al. 2013) and the ssj500k corpus. Moreover, the programs that implement individual modules are, by now, rather outdated. At the very least, TnT should be replaced by a newer tagger, which should be open source and system independent.

As mentioned earlier, normalisation in the context of modernising historical Slovene words has already been implemented in ToTrTaLe. However, the rules were written manually, and since their implementation automatically trained normalisation models using character-based statistical machine translation (Scherrer and Erjavec 2013) have been shown to perform better. A standard tool that can implement this method is Moses (Koehn et al. 2005), a statistical machine translation system that was, for the task of modernising Slovene words, trained on word pairs of a historical (non-standard) word and its modernised (normalised) version. This approach is useful not only for modernising historical words, but also for standardising contemporary texts with non-standard orthography, such as texts in computer mediated communication. Character-based statistical machine translation has already been tested for standardising words in Slovene tweets (Ljubešić et al. 2014), and has produced promising results. An issue related to normalisation is which texts to normalise: if normalisation is also used on texts with standard orthography, it is likely that completely standard words would be "normalised" as well, doing more harm than good. We have trained a system that uses machine learning on a small sample of Slovene tweets and other user-generated content from the Web, all manually annotated with their level of standardness to estimate (and annotate) how non-standard new texts are (Ljubešić et al. 2015). Normalisation could then be used only on texts that have been automatically annotated as non-standard.

2.3 Obeliks tagger and parser for Slovene

As part of the "Communication in Slovene" project, a tool called Obeliks (Grčar et al. 2012) was also developed. As with ToTaLe, the tool tokenises the input text, segments it into sentences, adds morphosyntactic tags and lemmatises it. It uses a module with handwritten rules for tokenisation, a purpose-built machine

learning tool for morphosyntactic tagging, and the machine learning LemmaGen program (Juršič et al. 2010) for lemmatisation. The MSD tagger is special in terms of not relying solely on a model automatically generated from a training corpus, but also using handwritten expert rules, which filter hypotheses generated by the model, and combining the results of the lemmatiser and the tagger, assuring that they are not contradictory. Obeliks has been trained on a manually annotated corpus (Arhar 2009; Krek et al. 2013c), and gives the best results among those tools for Slovene that are publicly accessible. At the moment, the tagger's main problem is probably its implementation in the C# programming language, which is designed to work on Windows and cannot be easily used on other platforms, such as Linux.

Obeliks was also used to annotate the Gos corpus of spoken Slovene (Verdonik and Zwitter Vitez 2011), the KoRP corpus of public relations texts (Logar 2013), the Šolar developmental corpus (Rozman et al. 2012) and the Gigafida corpus; the annotations from Gigafida (as well as the texts themselves) are also part of the KRES, ccGigafida and ccKRES corpora (Erjavec and Logar 2012).

A parser for Slovene (Dobrovoljc et al. 2012) was also built within the abovementioned project, where the well-known MSTParser dependency parser (McDonald et al. 2006) was trained on the dependency annotated part of the ssj500k corpus. The parser gives relatively good results, but – as is common for any linguistic annotation, especially parsing – its accuracy depends heavily on the genre of the text – the more the genre differs from that of training dataset, the poorer the results. An evaluation of the parser showed that its accuracy also depends substantially on the type of dependency relation, since this ranges from 54% to 96%.

2.4 Other tools

Named entity recognition (NER) for Slovene is supported by two tools. An NER tagger was developed (Štajner et al. 2012) which uses machine learning based on conditional random fields, with the model trained on ssj500k. The tool is available under an open license, but is rather difficult to use since its installation and use are relatively poorly documented. The second tool (Ljubešić et al. 2013) is based on StanfordNER (Finkel et al. 2005), which also works with conditional random fields and was also trained on ssj500k, but in combination with the slWaC corpus of the Slovene Web (Ljubešić and Erjavec 2011). The latter allows the tool to collect more accurate information on features and their distribution, which proves to be very efficient in decreasing the number of false positives as well as improving recall. The models are openly accessible, and StanfordNER is well maintained and documented.

Term extraction for Slovene has been implemented and studied through a number of experiments (Logar and Vintar 2008; Vintar 2009; 2010; Logar et al. 2013), yet these are not available under an open licence nor maintained. The tools are mostly based on a combination of linguistic knowledge about terms (especially which patterns of MSD tags can represent terms) and mathematical knowledge about the distributional features of word sequences in corpora. Identifying terms by using machine learning methods has not been tried yet for Slovene, and there are also no openly accessible training sets that could be used for this purpose.

3 GUIDELINES FOR THE FUTURE ANNOTATION OF CORPORA

3.1 Improving annotation schemes

Before focusing on how to improve the tools or corpora used for training, it is necessary to discuss annotation schemes that (manually) annotated corpora are based on. The design of these schemes should be re-thought and tested to improve the accuracy of the tools, while also preserving or even improving the linguistic informativeness of the individual levels of annotation.

Grammatical information about individual words from corpora such as ssj500k, Gigafida, and KRES, as well as from the morphosyntactic lexicon Sloleks, is based on the morphosyntactic specifications developed in the project JOS "Linguistic Annotation of Slovene" (Erjavec and Krek 2008). This system originates from and is in line with the MULTEXT specifications (Ide and Véronis 1994), or its subcategory MULTEXT-East. The MULTEXT-East 4.0 specifications (Erjavec 2012) cover 12 languages, including almost all Slavic languages, and are, for Slovene, identical to the JOS specifications.

The JOS specifications define 12 parts of speech: noun, adjective, verb, adverb, pronoun, numeral, preposition, conjunction, particle, interjection, abbreviation and residual. The majority of these contain information on their morphosyntactic features, either lexical ones, such as is the noun common or proper or the verb auxiliary or main, and inflectional ones, such as number or case. All valid combinations of a part of speech and its features and encoded as strings (MSD tags), where each position in the string represents a certain attribute; its value is expressed through a one-letter alphabetic character. For example, the meaning of the string *Ncmsn* is *part of speech = Noun, type = common, gender = masculine, number = singular, case = nominative.* String encodings as well as features (i.e. attributes and their values) are available both in Slovene and in English. JOS comprises 1,902 different MSD tags, which are listed in the specifications together

with corpus examples. MSD tags, such as *Ncmsn*, are then used in morphosyntactically annotated corpora, and also in the morphosyntactic lexicon Sloleks to define paradigms of word forms for individual words.

JOS tags are used in many corpora, but the full set of the tags and their features may not be the most suitable for all applications. It is possible to opt for a pruned tagset, excluding features where morphosyntactic taggers are most error-prone (e.g. grammatical case), or all inflectional features if lexical features suffice for the purposes of the project. Such alternatives, which reduce the size of the JOS tagset and increase the accuracy of taggers, have already been used: a detailed study in Krek (2011) suggested several options on how to reduce the tagset, while Erjavec (2013) reduced the set to 32 tags, which are limited to the part of speech and some of its lexical features. However, more studies would be needed to determine what the optimal tagset for each specific purpose would be.

More recently another interesting possibility has appeared, as with the ongoing Universal Dependencies project (Nivre et al. 2105) specifications and treebanks for many languages are being developed, including for Slovene. In addition to defining syntactic relations, the project offers a universal set of morphosyntactic features (with optional language-specific extensions). Although the drive toward universality inevitably leads to lower adaptability of the scheme to individual languages, its subsequent comparability between many languages may outweigh individual cases of poor performance.

There is an even greater need for additional studies on the set of syntactic tags and relations from the JOS and "Communication in Slovene" projects (Erjavec et al. 2010; Arhar 2009), since these have not yet been thoroughly tested. One of the issues is parses with multiple roots, which pose a problem for automatic parsers (Javoršek 2015). However, both theoretical and practical recommendations from the Universal Dependencies project should also be taken into account in the further development of a syntactic annotation system.

Current categories for named entities as used in the ssj500k corpus have also proven deficient. Štajner et al. (2012) showed that by dividing the "other" category (i.e. for those named entities that are neither personal nor geographical names) into names of organisations and "other" not only gives a more fine-grained set of categories, but also improves the overall quality of annotation. This conclusion is in line with findings from the authors of the Czech corpus of named entities CNCEC, which has no less than 62 categories of named entities (Ševčíková et al. 2007). Here it was found that reducing the number of named entity categories also led to worse results in annotation. Therefore, further increases in the number of named entity categories should also be considered for Slovene.

3.2 Improving the accuracy of tools

The tools described above differ in the quality of annotation as well as their ease of use. It would be useful to increase the accuracy of annotation in all of them, since every error is problematic in two respects. On the one hand, low precision brings noise to the dataset, since lexicographers also obtain incorrect results to their queries. For example, when querying a certain lemma, the words that were incorrectly annotated with this lemma will distort the overall picture regarding concordances, collocations, word sketches etc. However, here the lexicographer can at least go through the examples, decide which are correct, and discard the rest, a time-consuming but doable task. The second issue is low recall, which is worse. In this case, lexicographers cannot obtain some of the results they are interested in, because the tools fail to discover them: if a word is completely or mostly incorrectly lemmatised, it will not be found when searching its lemma or there will be few results. The main goal of automatic corpus annotation should thus be improving both the accuracy and recall in all processing steps – especially the initial ones (tokenisation, MSD tagging, lemmatisation), since every error gets multiplied further down the processing chain, making lexicographic analysis much more difficult to carry out.

It is becoming obvious for most annotation levels that better and quicker results are achieved through machine learning rather than with handwritten rules. But machine learning requires high quality manual annotated datasets for training, and such datasets are also needed for testing the quality of the tools, regardless of whether they use machine learning methods or handwritten rules. To improve the quality of the tools, it would thus be useful to increase the size as well as the diversity of manually annotated corpora. Here, it would not be necessary to annotate entire texts – methods of active learning could pick out examples that would be most useful in helping improve the trained model. Depending on the level of annotation, it would make sense to also expand supporting data sources, particularly lexicons and lexical databases, since these can offer linguistic information in a refined form.

Also noteworthy is the conceptual model of annotation or expanding support sources that build on a "virtuous circle": additional manually annotated corpora train the programs for better annotation, which results in a better basis for the next cycle of manual annotation, and this circle or rather spiral can be repeated multiple times.

The accuracy at all annotation levels heavily depends on the tokenizer, since its errors are transferred into all further annotation steps, while the errors in tokenisation directly block the possibility of finding incorrectly tokenised words.

Therefore, a lot of effort has already been put into building a reference tokenizer for Slovene (Krek 2011), which is, to an extent, already implemented in Obeliks, although its functioning could be further improved. For example, the tokenizer does not recognize the already mentioned "*ga.*" as an abbreviation in sentences such as "*Spoštovana ga. Micka!*" (*Dear Mrs. Mary!*). But one also needs to be aware of the fact that every change to a tokeniser that was previously used to produce existing (also manually) annotated corpora results in incompatible resources, which has a negative effect on annotation as well as on extracting grammatical information from corpora. Such cases come to light when, for example, corpora annotated with ToTaLe are used together with those annotated with Obeliks. A study of Web-specific vocabulary, where keywords of slWaC in comparison with the KRES reference corpus were examined, has found many "key words" to be exactly those that are tokenised differently in ToTaLe versus Obeliks (Erjavec and Ljubešić 2014).

When improving morphosyntactic tagging, it is useful to carry out experiments on which methods or combinations of methods truly generate the best results. It has already been shown, for example, that the use of meta-learning, which combines the results of Amebis' rule-based tagger and the TnT statistics-based tagger, gave better results than either tool used separately (Rupnik et al. 2010).

3.3 Improving the technical side of tools

In addition to improving the accuracy of the tools, other technical improvements could also be made, i.e. simplifying their installation and use, as well as improving the ease of their integration.

A general recommendation is to use open source tools that are independent of the language and the computer platform, are based on machine learning, well documented, and maintained on one of the platforms for revision control, such as Git, which have an active group of developers and users that can communicate through a forum, report errors or send suggestions for improvements. Two examples of such platforms are Moses and – to some degree – StanfordNER. Even though purpose-built tools for Slovene would have the advantage of being better fitted to the specifics of the language (or its theoretical linguistic framework), it is arguable whether this outweighs the amount of work needed for their development and maintenance. Such tools may produce good results at a certain stage in their development, but it is very likely that continued progress in machine learning will bring increasingly better results. It is therefore more reasonable to put the effort into developing annotated corpora of Slovene that can serve as quality training sets rather than into complex rule-based tools built solely for Slovene.

One possible approach to building annotated corpora for lexicography is deciding that the accessibility of tools is irrelevant as long as they are successfully used for corpus annotation, yet this makes expanding and also maintaining the corpora more difficult. Additionally, it would not be possible to use these tools for other purposes nor annotate other corpora that are not related to this particular lexicographic project or study. Using closed and proprietary tools also prevents the results of annotation from being checked or reproduced.

The next issue is the connectivity of individual tools, both regarding their import and export formats as well as their limitation to certain computer platforms. The use of the above-mentioned implementation of Obeliks, currently the best openly-accessible MSD tagger for Slovene, is limited to the Windows OS. This makes it incompatible with the Linux environment, which is traditionally much better equipped with open source taggers and other tools. However, there is a growing number of platforms which enable users to set up and run online workflows, such as WebLicht (Hinrichs et al. 2010). These systems implement individual programs or modules in such a way that they run as Web-based services, possibly in computer clusters, while the execution of the workflow as a whole (e.g. tokenisation → MSD tagging → lemmatisation) is controlled by a central server, which calls these Web services as specified by the workflow. Perhaps this model of annotation is where the future lies, but the current solution – especially for processing large corpora – is still execution on local computers that are connected into clusters and typically possess large processing as well as memory capacities. It is thus crucial for annotation tools to be independent of the operating system or platform on which they are being executed. In practical terms, this requires them to be written in one of the standard open source programming languages, such as Java or Python.

Besides platform independence, compatibility of import and export formats needs to be specified, e.g. as done WebLicht; more on this in section 4.

Another interesting challenge for scientists and developers is the architecture of system for text annotation. Most current implementations function by choosing the best candidate at every step of annotation. Yet the best candidate from one step may turn out to be the wrong choice when more information becomes available, as only later steps in the processing chain could correctly disambiguate among potential candidates. For example, only when taking into consideration the syntax can the system determine that the first "*ga.*" from "*Videl sem ga. Micka ga je tudi videla.*" is not an abbreviation but a pronoun followed by a period that marks the end of the sentence. A newer trend in this field are systems that use Bayesian networks (Finkel et al. 2006) instead of a simple pipeline of taggers. In the former, each tagger represents one variable of the system, allowing it to make approximate assumptions that determine the best tags globally.

3.4 Proposed chain of annotation tools

To annotate corpora of Slovene language for lexicographical purposes, only those levels of annotation were chosen where existing tools are already readily available, rather than all possible or potentially useful ones that are yet to be developed. The following paragraphs describe a suggestion for a chain of annotation tools. Out of all tools presented above only fully open source tools, both in terms of software and models of the Slovene language, have been chosen. For each tool some fairly simple suggestions for improvement are given.

- **Obeliks**: tokenisation, MSD tagging and lemmatisation. It would be useful to implement the systems in one of the standard programming language and re-train its morphosyntactic and lemmatisation models. Word normalisation could be added instead of having it implemented in a separate module.
- **Moses**: normalisation of word forms. The program should support several normalisation models, at least one for modern non-standard Slovene and one for historical Slovene. If the text needs to be normalised and, if so, which model to use could be either decided automatically according to the content or based on the metadata of the text.
- **MSTParser**: shallow parsing. The existing model for syntactic analysis could be used, but it would be useful to implement a conversion from the JOS scheme to the Slovene version of Universal Dependencies and train the parser on the latter, too. Corrections in certain parts of the training corpus ssj500k would be useful, as well as increasing the size of genres that are currently poorly represented but seem to be syntactically different from those already in the corpus. Experiments could also be made with some other, more contemporary parsers to see if better results could be obtained.
- **StanfordNER**: named entity recognition. The size of the dataset used for training (at the moment only ssj500k) could be increased and, more importantly, made more heterogeneous.
- As mentioned, there is still no maintained and openly accessible tools for annotating terminology, which is why term extraction should probably be programmed from scratch. The existing patters of morphosyntactic tags that represent potential terms could also prove useful.

One question is still open: how to link the above-listed tools, which are quite heterogeneous. For efficient automatic annotation of large corpora, the best solution is the installation and parallelisation of a chain of taggers on high capacity

Linux servers or clusters of such servers, where the conversion of their input and output formats should be implemented in such a way that they are compatible. One possibility is directly using the TEI format, but a stand-off format would be more appropriate to make the taggers work more efficiently; more on this in the following section.

4 ANNOTATION FORMAT FOR CORPORA

The structure of a corpus may be very complex, both regarding its metadata as well as linguistic annotations. Slovene corpora mostly follow the TEI guidelines (TEI 2013), which cover nearly all the above-mentioned annotation levels, as well as some others. The guidelines are well maintained under the auspices of the international TEI Consortium. The TEI format is also supported by a plethora of tools for building custom-made XML schemas and converting from and into various formats, such as from Word to TEI or from TEI to HMTL. The names of TEI elements have been translated into Slovene and are being applied by a number of users in the field of digital humanities (Erjavec et al. 2004; Ogrin et al. 2013).

In TEI the majority of linguistic tags are written directly as XML elements, using e.g. <w> for a word and <name> for a name. The advantages of such an in-line approach are the transparency of elements and simple formal validation of the format; the tags as well as the text can both be simply corrected. This solution has also several weaknesses: the elements need to be correctly nested (XML primarily supports tree structures), and with an increasing number of elements the XML becomes more difficult to understand and control. Moreover, the files with in-line elements can become quite large. These are the reasons why annotation schemas intended primarily for automatic annotation more often use the stand-off approach, where the base text remains unchanged. Instead, the annotations generated by individual tools point to the corresponding parts in the text or one of the element layers. Such an approach is used by the above-mentioned WebLicht (Hinrichs et al. 2010), which has built a shared corpus format called TCF. The same approach is also defined by the MAF standard, which is used to annotate morphosyntax (ISO 24611, 2012).

Although the stand-off approach is technically simpler and offers greater flexibility, it makes it harder to discover errors and link together individual annotation levels. Furthermore, the data to which the annotations point to must not be altered, or else the pointers become invalid. This becomes problematic when the text itself or (some of) the elements would need to be corrected, either manually or semi-automatically. The TEI format is thus primarily suitable for manually

annotated reference corpora, where it is crucial to have the tags and the corpus format as thoroughly checked as possible.

Most corpora mentioned in this monograph are TEI compliant, but the use of its recommendations is complex. What is more, it is now over two decades since the compilation of some Slovene corpora, and the TEI guidelines have been since then modified a number of times. When adding new annotations, some past decisions may prove hard to generalise. Therefore, it would not only be useful to annotate existing corpora with new tools and models, but also to standardise their encoding, which could then serve as a reference for the corpora needed for a new dictionary of modern Slovene.

5 CONCLUSION

This chapter has provided an overview of the levels of automatic linguistic annotation that should be part of the annotation of corpora to be used as the basis for lexicographic analyses of modern Slovene, as well as for other purposes. We have given an overview of existing research in this field and then focused on a concrete set of open source and mainly language independent tools and their models for Slovene, and provided suggestions for their improvement. A short description of the proposed corpus encoding has also been provided.

Annotations in large corpora are always assigned automatically, which is why users of need to be aware of the fact that such tools will inevitably make errors, resulting in poorer performance with regard to extracting information that is relevant for lexicographical or similar purposes. Further improving the accuracy of these tools thus remains a priority.