Implementing an End-to-End Treebank-Informed Pipeline for Bulgarian

Alexander Popov
AIaLT
IICT-BAS, Bulgaria

Petya Osenova
AIaLT
IICT-BAS, Bulgaria

Kiril Simov
AIaLT
IICT-BAS, Bulgaria

`{alex.popov|petya|kivs}@bultreebank.org`

Abstract

The paper reports on the implementation of an NLP pipeline for Bulgarian, developed within the spaCy framework and based on BulTreeBank as main source for training and test data. This new end-to-end pipeline aims to ensure easier technical maintenance and synchronization between modules, superior processing speeds – for use in real applications, and greater flexibility of adaptation. We discuss the challenges encountered in the implementation process and the solutions adopted, including the architecture itself, as well as its quantitative and qualitative evaluation.

1 Introduction

Implementing a pipeline solution for processing a specific language is a task characterised by non-trivial challenges. Different implementation frameworks necessitate decisions about the overall architecture, which in turn constrain the possible solutions to concrete tasks.

A previous end-to-end pipeline for Bulgarian has been implemented in Java and in the XML-based CLaRK system[1]. It includes the following modules: tokenizer, sentence boundary detection, POS tagger, lemmatizer, and a transition-based dependency parser. The tokenization, sentence segmentation and lemmatization modules are implemented as rules in the CLaRK system, while POS tagging and dependency parsing are carried out by statistical models trained on annotated data. The Mate Tools[2] software has been used for training the models, where the training data is an older version of the BTB treebank, which includes fewer sentences and differs in annotation types from the Universal Dependencies (UD) version. Later versions of this pipeline include additional processing modules (e.g. word sense disambiguation), which however are not trained on annotated data and are implemented in other frameworks (Simov et al., 2016).

Currently there is no freely-available end-to-end processing pipeline for Bulgarian that meets the following criteria:

- is implemented within a single framework;
- includes semantic analysis capabilities (word sense disambiguation, named entity recognition);
- achieves competitive accuracy scores;
- affords processing speeds suitable for real applications;
- can handle big volumes of data.

This short paper reports on the implementation process of such a pipeline. The creation of a language processor for Bulgarian is inevitably related to the Bulgarian treebank — BulTreeBank (BTB), a version of which is freely available through the Universal Dependencies initiative. Since the treebank has been annotated with named entities, POS labels and features, as well as converted to syntactic dependencies, it is used as the main training data for the pipeline modules. We have also obtained the current version of the BTB-WordNet, which has made it possible to add a module for semantic analysis as well.

The accuracy scores reported in the paper are generally lower than those achieved with the older pipeline. However, the training and evaluation data has evolved significantly over time, making a

[1]http://bultreebank.org/en/clark/bulgarian-nlp-pipeline-in-clark-system/
[2]`https://www.ims.uni-stuttgart.de/en/research/resources/tools/matetools/`

fair comparison difficult. The new pipeline modules have been trained on the current version of UD-BulTreeBank that – compared to previous versions – consists of more sentences and more varied syntactic structures. Thus, accuracy has seemingly dropped, but the pipeline can actually handle more challenging syntax. In addition to that, the current system, which is a work in progress, has the advantages that it is much more robust, faster, intuitive, and extensible, making it suitable for practical applications.

2 Implementation of the pipeline

For the implementation of the Bulgarian processing pipeline, the spaCy framework for NLP was chosen – thereby allowing us to implement the pipeline entirely in Python code. spaCy has been developed to be suitable for industrial-strength solutions, which means that it is fast, well-structured, flexible, and easy to use. Comparison with other popular frameworks shows that it works at significantly greater speeds.[3] It also offers a variety of NLP tools that can be easily added to or removed from the pipeline: from tokenization and sentence splitting, to dependency parsing and entity linking. spaCy models trained on data for English and other languages consistently achieve accuracy scores that are close to the state of the art. One potential disadvantage of spaCy is that its neural architectures are fixed in advance and there is not much freedom in configuring different models. However, due to spaCy's modularity and non-destructive tokenization (i.e. the original input can always be recovered from the processed output), it can be combined relatively easily with standard deep learning frameworks.

Adapting spaCy to a new language includes two major steps: 1) adding language-specific lists and rules for tokenization and lemmatization;[4] 2) training statistical models on language-specific data. While model training on available data is largely a seamless process, the first step depends more heavily on the specificities of the particular language.

2.1 Rule-based and dictionary-based language-specific processing

2.1.1 Tokenization

Tokenization is carried out via rules and language-specific exceptions. This amounts to compiling:

- a list of strings, each one of which is to be analyzed a single token;
- attributes associated with abbreviated tokens: lemmas, as well as morphological analyses;
- regular expressions for handling tokens with special symbols, like hyphens, apostrophes, etc.;
- regular expressions for handling punctuation marks that should not split strings into tokens (e.g. if we want to analyze dates in the [DD.MM.YYYY] format into single tokens).

A list of tokens, including additional information about lemmas and morphology, was compiled on the basis of a focused lexicon; regular expression cascades were iteratively devised during development.

2.1.2 Lemmatization

Bulgarian is a relatively rich language in terms of its morphology – the unique fine-grained morphological tags in the Bulgarian treebank number 578.[5] This means that a high ambiguity of word forms can be expected, and simple part-of-speech tags (e.g. the Universal POS tagset (Petrov et al., 2011)) will not always be enough to disambiguate the correct lemma per word form. For instance, the same surface verb forms may have different underlying lemmas depending on how they are used. Consider the example in Table 1. The simple POS tag is the same for both cases and would therefore be insufficient for correct disambiguation. Knowing that one word form belongs to an impersonal verb, and the other does not, however, would be sufficient information – which is indeed provided by the full morphological tag.

Thus, a language-specific morphological dictionary was constructed, on the basis of an existing grammatical resource, and added to the lookup tables that the spaCy language model accesses for lemmatization. The Bulgarian lemmatizer takes a token string (i.e. word form) and attempts to pair it with the predicted morphological tag as a joint dictionary key, so that it can extract the correct lemma from the dictionary. In order to ameliorate errors from the POS tagger, a fallback option is included in the logic of

[3] `https://spacy.io/usage/facts-figures`

[4] While it is true that data-driven approaches can also lead to excellent results (Kondratyuk et al., 2018), we have chosen to follow the spaCy schema for developing language models.

[5] http://bultreebank.org/wp-content/uploads/2017/04/BTB-TR03.pdf

word form	Gloss	POS	Morpho-tag	Lemma
вървя	“luck was on somebody’s side”	VERB	Vniif-o3s	върви-ми
вървя	“he/she/it walked”	VERB	Vpiif-o3s	вървя

Table 1: Comparison of verb forms.

the lemmatizer: in case no matches for the word form/morpho-tag pair are found, the simple POS tag is used to obtain possible (the default logic in spaCy); the word form/POS mappings are bundled in a separate lookup table. Currently the lemmatizer selects the first candidate in the list of lemmas matching the word form/POS pair, which would produce errors in some cases (due to the aforementioned ambiguity); one planned improvement is to order the list of lemma candidates using frequency distributions. In the cases when neither the morphological tag nor the POS tag yield an associated lemma, a simple lookup from word form to lemma is used. If that strategy is also unsuccessful, a lowercased version of the string is returned.

2.2 Statistical models for NLP

2.2.1 POS Tagging and Dependency Parsing

Training models on annotated data with spaCy’s neural architectures is a straightforward process. In our case, we have used the Universal Dependencies files for training the POS tagger, dependency parser (including labeled attachment), and sentence splitter, which depends on the parser module to correctly segment sentences.[6] We present the accuracy results on the development and test sets in the next section.

2.2.2 Named entity recognition

BulTreeBank has been annotated with information about Person, Location and Organization entities, but since NER data is not included in the UD version, we trained the NER module separately, after the POS tagger and parser. We also included data from the BSNLP corpus (Marinova et al., 2020), which has been originally compiled for a special task on NER and includes Event, Product and Other types, in addition to Person, Location and Organization. The two corpora were processed into the spaCy-readable IOB format, concatenated and shuffled, to balance them between the training (20803 sentences) and development (2312 sentences) portions. We provide evaluation metrics for NER in the next section.

2.2.3 Word sense disambiguation

The final module included in the pipeline carries out word sense disambiguation (WSD). The spaCy framework does not provide off-the-shelf WSD functionality, therefore a different solution had to be adapted. The EWISER system (Bevilacqua and Navigli, 2020) was chosen, due to several considerations: superior accuracy compared to other systems, easy integration with spaCy, and multilingual support.

EWISER improves the state-of-the-art in WSD on the popular evaluation framework for English (Raganato et al., 2017), “breaking through the 80% glass ceiling”. It accomplishes that via a novel approach to representing word senses in relation to the other senses to which they are related through the WordNet semantic network. EWISER is a neural network classifier that uses the powerful BERT model (Devlin et al., 2018) for contexualized embeddings to construct the context representation for each input word. It also uses a synset embedding matrix to transform the final hidden layer of the network into a vocabulary-sized vector – thus, instead of randomly initializing the final linear transformation matrix, structured knowledge about senses is added into the calculation of the logits. The logits are additionally “structured” by infusing information about the relations between all synsets in the vocabulary. Apart from achieving the highest reported results on the standard English datasets, EWISER also fares well when tested against data for other languages. Even more impressive is the fact that it achieves state-of-the-art results on German, French, Italian and Spanish data when trained on the SemCor corpus for English and using the multilingual BERT model. The EWISER implementation[7] provides a special class for initial-

[6]`https://universaldependencies.org/treebanks/bg_btb/`
[7]`https://github.com/SapienzaNLP/ewiser`

izing the system as a spaCy module, which can then be directly added to a pipeline; it also provides the relevant models for processing multilingual data.

Currently, an enriched version (around 25,000 synsets) of the Bulgarian BTB-WordNet (Osenova and Simov, 2018) is being prepared for official release, which involves consolidating the indices and ensuring their correct mapping to the Princeton WordNet (PWN), version 3.1, as well as the actualization of the BulTreeBank sense-annotated data with the updated IDs. When this work is done, it will be possible to properly train and evaluate an EWISER model on Bulgarian data. As of now, we have only been able to adapt the multilingual model for processing Bulgarian text, and thus provide only a qualitative analysis in the next section. The adaptation itself comprises of a chain of transformations:

- preparing a dictionary that maps lemmas to possible synsets;
- replacing the Bulgarian synset IDs with IDs from PWN;
- mapping the PWN IDs from version 3.1 (used in BTB) to version 3.0 (used in EWISER);
- mapping PWN IDs to BabelNet IDs, in order to produce the final dictionary required by EWISER;
- compiling a list of lemmas that the system can recognize (i.e. they are present in the dictionary).

After that, the spaCy EWISER module can be run on Bulgarian text, with the sense annotations available from the **token._** attribute. In addition to training a model on Bulgarian data, we plan to improve this part of the pipeline by experimenting with custom-made synset embeddings.

3 Evaluation

POS tagging and dependency parsing The POS tagger achieves accuracy of 94.13 % on the development set and 94.49 % on the test set. The metrics for the parser are as follows: 83.03 % for LAS and 88.95 % for UAS on the development data, and 83.95 % LAS / 89.71 % UAS on the test data. In the multilingual setting of the UD parsing shared task, the best model achieved LAS 91.22 % on the Bulgarian treebank (Zeman et al., 2018). However, there is evidence that UD-trained models perform better in multilingual and cross-lingual settings, compared to monolingual ones (Smith et al., 2018). The average reported accuracy across the models in the shared task aligns with the results here.

The detailed results for the UD relations are given in Tables 2, 3 and 4. Only three relations have not been evaluated: **dep**, **vocative** and **appos**. The nominal subject (**nsubj**) and direct object (**obj**) relations get the best results in table 2, as can be expected. On the other hand, clausal subjects (**csubj**) and clausal complements, namely the infinitive-like type (**xcomp**), are hard to detect. The most difficult case is the passive sentential subject (**csubj:pass**). This is also expected, because it is not marked in any special way and depends on the passive word form of the verb.

UD relation	**P**	**R**	**F**	**UD relation**	**P**	**R**	**F**
obj	80.75	76.86	78.75	iobj	61.89	60.65	61.26
csubj	57.57	45.23	50.66	nsubj	81.07	77.61	79.30
ccomp	79.43	57.04	66.40	xcomp	40.0	80.95	53.54
nsubj:pass	60.71	75.55	67.32	csubj:pass	40.0	25.0	30.76

Table 2: Evaluation of the UD core relations.

Table 3 shows that the parser achieves its highest scores in parsing relations that have fixed word order positions in Bulgarian phrases. These are prepositional phrases (**case**), determiners (**det**), adjectival modifiers (**amod**), numeral modifiers (**nummod**), expletives (**expl**). In contrast, relations that are expressible in a variety of ways are scored lower. These are: the oblique relation (**obl**), secondary predication (**acl**) and adverbial clauses (**advcl**), among others.

The highest-scored relation in table 4 is the one which includes the root node, followed by **cc** and **fixed**. Verbless sentences are clearly more challenging, the most difficult relation being that of **parataxis**.

Named entity recognition The combined (cross-category) results for NER are as follows: precision – 92.75 %; recall – 93.31 %; F-measure – 93.03 %. The detailed results from the NER module, the first to be reported on this combination of data, are presented in table 5. The model produces the most

UD relation	P	R	F	UD relation	P	R	F
mark	89.05	89.5	89.27	expl	96.01	92.8	94.37
advcl	69.09	65.8	67.45	discourse	73.68	63.63	68.29
cop	68.18	83.33	75.0	case	96.01	96.57	96.29
aux	94.18	83.21	88.36	det	94.51	94.51	94.51
nmod	79.36	79.52	79.44	amod	93.24	93.54	93.39
advmod	81.53	81.76	81.65	obl	65.29	57.48	61.14
nummod	91.42	91.42	91.42	aux:pass	43.51	91.93	59.06
acl	60.0	53.33	56.47	acl:relcl	73.10	76.31	74.67

Table 3: Evaluation of the non-core UD relations.

UD relation	P	R	F	UD relation	P	R	F
root	89.97	90.13	90.05	cc	89.50	89.15	89.32
conj	66.93	70.65	68.74	fixed	92.68	79.16	85.39
flat	65.07	90.44	75.69	parataxis	9.09	54.54	15.5

Table 4: Evaluation of the other UD relations.

accurate annotations for the event (EVT) class. This is due to the fact that the additional training data (BSNLP) focuses mainly on a single event – Brexit. The identification of locations (LOC) also achieves good results, while those for person (PER) and organization (ORG) are slightly lower – mostly due to the frequent occurrence of regular polysemy and the partial identification of chunks; the model also seems to have a bias toward words with capital letters at the beginning of sentences; it is worth noting that the PER category includes etnonyms in addition to names. The recognition of products (PRO) fares well, while OTH (other), being a catch-all category, has significantly lower recall, and thus – lower F-measure.

Word sense disambiguation The WSD module analysed only content words that can be linked to possible meanings from the Princeton WordNet. It relied on the lemmatizer and POS tagger in order to retrieve the relevant word senses from the dictionary. Here is an example:

Това е първата ми реакция.
(This is my first reaction)

The numeral and the noun are annotated with the following lemma/POS/sense information:

- първата първи ADJ wn:01010862a: preceding all others in time or space or degree
- реакция реакция NOUN wn:00863513n: an automatic instinctive unlearned reaction to a stimulus

For the moment we do not have any automatic evaluation of the WSD module, and only qualitative observations can be made regarding the output of the model. Nouns and verbs in one hundred sentences were checked for the assigned senses. One observable problem are lexemes which have not been mapped to senses – because they are missing in the dictionary, or because of incorrect lemma/POS annotations. Another problematic issue are multi-word expressions — which need to be processed as single tokens by the pipeline in order to be correctly disambiguated. For example, in the MWE "изпускам си нервите" (leave-I REFL nerves) ('lose one's temper'), the system identified the lemma 'nerve', but not the verb 'leave'. In some cases there is a bias towards one sense of the lexeme, while the rest are ignored. For example, in one and the same sentence the lexeme "скелет" (skeleton) is used in two different senses: "the hard structure (bones and cartilages) that provides a frame for the body of an animal", and "something reduced to its minimal form", but the system assigns the first sense in both cases. This might be due to a skewed distribution of samples in the training data (in this case — SemCor).

4 Conclusion

The paper reports on the initial stage in the implementation of an end-to-end pipeline for Bulgarian, using the spaCy framework. The pipeline comprises of rule-based (tokenizer, lemmatizer) and statistical

NER lables	P	R	F	NER lables	P	R	F
PER	91.57	91.47	91.52	EVT	98.73	97.90	98.31
ORG	90.91	93.41	92.14	LOC	95.25	96.87	96.05
PRO	89.36	89.36	89.36	OTH	76.36	56.0	64.61

Table 5: Evaluation of the NER categories.

(POS tagging, NER, UD parsing, and WSD) components, working together in a shared setting. The initial evaluation results provide a promising baseline for future improvements, which would focus on: better tokenization through more precise rules and syntactic parses; better mapping between POS tags and potential lemma candidates; adapting the BTB-Wordnet data for training WSD models; adding more gold data for the training of the NER module and the UD parser.

Acknowledgements

This work was partially supported by *the Bulgarian National Interdisciplinary Research e-Infrastructure for Resources and Technologies in favor of the Bulgarian Language and Cultural Heritage, part of the EU infrastructures CLARIN and DARIAH – CLaDA-BG*, Grant number DO01-272/16.12.2019. Alexander Popov's contribution has been supported by the Bulgarian Ministry of Education and Science under *the National Research Programme "Young scientists and postdoctoral students"* approved by DCM # 577 / 17.08.2018.

References

Michele Bevilacqua and Roberto Navigli. 2020. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. Lemmatag: Jointly tagging and lemmatizing for morphologically-rich languages with brnns. *arXiv preprint arXiv:1808.03703*.

Iva Marinova, Laska Laskova, Petya Osenova, Kiril Simov, and Alexander Popov. 2020. Reconstructing ner corpora: a case study on bulgarian. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4647–4652.

Petya Osenova and Kiril Simov. 2018. The data-driven bulgarian wordnet: Btbwn. *Cognitive Studies*, 18.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110.

Kiril Simov, Petya Osenova, and Alexander Popov. 2016. Towards semantic-based hybrid machine translation between bulgarian and english. In *Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation (SedMT 2016)*, pages 22–26.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal Dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium, October. Association for Computational Linguistics.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium, October. Association for Computational Linguistics.