ГОДИШНИК НА СОФИЙСКИЯ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Том 110

ANNUAL OF SOFIA UNIVERSITY “ST. KLIMENT OHRIDSKI”
FACULTY OF MATHEMATICS AND INFORMATICS
Volume 110

AN IMPROVED BULGARIAN NATURAL LANGUAGE PROCESSING PIPELINE

MELANIA BERBATOVA AND FILIP IVANOV

In this paper, we present a language pipeline for processing Bulgarian language data. The pipeline consists of the following steps: tokenization, sentence splitting, part-of-speech tagging, dependency parsing, named entity recognition, lemmatization, and word sense disambiguation. The first two components are based on rules and lists of words specific to the Bulgarian language, while the rest of the components use machine learning algorithms trained on universal dependency data and pretrained word vectors. The pipeline is implemented in the Python library spaCy (https://spacy.io/) and achieves significant results on all the included subtasks. The pipeline is open source and is available on Github (https://github.com/melaniab/spacy-pipeline-bg/) for use by researchers and developers for a variety of natural language processing and text analysis tasks.

Keywords: natural language processing, language pipeline, word sense disambiguation

CCS Concepts:

• Applied computing∼Document management and text processing∼Document capture∼Document analysis

1. Introduction

A language pipeline consists of a sequence of steps targeted towards processing and analyzing natural language data. A typical language pipeline might include steps such as tokenization, part-of-speech tagging, parsing, and semantic analysis among others. These steps are used as a preprocessing stage in many different tasks and applications that involve analyzing human language.

Large number of language pipelines are built with the Python natural language processing library spaCy, which offers easy and flexible way to create such systems. A spaCy pipeline can combine predefined components, such as tokenizer and part

DOI: 10.60063/GSU.FMI.110.37-50

of speech tagger, as well as custom developed components and other functions, depending on the end goal of the system.

There are previous works for building a Bulgarian pipeline based on previous versions of spaCy [18] or custom-built software [19]. However, since these works have been published, new neural-based based algorithms for tasks such as lemmatization have been put into practice, which improve the performance and also facilitate the automatic evaluation.

Currently in spaCy v.3, there are trained pipelines for more than 20 languages, including low-resource languages from the Balkans, such as Macedonian[1], Greek [17], and Romanian, but not Bulgarian. A language pipeline with good performance is crucial both for conducting research in the field of word processing and other related areas and for creating software applications for various purposes.

The goals of the current work are:

- to create an end-to-end, open-source pipeline for the Bulgarian language in spaCy v.3;
- to improve available lists of tokenizer exceptions and stop words and regular expressions for handling specific symbols and punctuation;
- to switch from rule-based to neural edit-tree lemmatization;
- to create custom modules for sentence splitting and for word sense disambiguation.

2. Related work

Savkov et al. [19] are the first to present a linguistic processing pipeline for Bulgarian including morphological analysis, lemmatization, and syntactic analysis of Bulgarian texts. Lemmatization and word sense disambiguation are performed by manually crafted rules, while part-of-speech tagging and morphological tagging are performed by tools based on support vector machines (SVMs). Different parts of the pipeline are developed as part of different systems, including the CLaRK system [20], Gtagger [6], and MaltParser [14].

Later, Popov et al. [18], present a spaCy-based pipeline consisting of modules for tokenization, lemmatization, POS tagging, dependency parsing, named entity recognition and dependency parsing. The lemmatization is still done by applying a set of rules over a large morphological dictionary, part of speech tagging, dependency parsing and named entity recognition are done by spaCy's built-in algorithms and word sense disambiguation is performed using the EWISER system [3]. There are some directions in which the pipeline can be improved. First, the pipeline is implemented in an outdated version of spaCy, it is not publicly available, and there is no discussion of the algorithms in use. Second, there is no information about the use of pretrained word vectors. And finally, there are no quantitative evaluation

[1] https://blog.netcetera.com/macedonian-spacy-f3c85484777f

results reported from the WSD system, and there is no analysis of the errors that the system produces.

3. Training data

3.1. Universal dependencies

The data in the Bulgarian treebank [16] consists of a total number of 11138 sentences, of which 8907 are in the train set, 1115 are in the development set, and 1116 are in the test set. These sets are formed in the following way: each first sentence is taken for the test set, each tenth one is taken for the development set, and the rest of the sentences form the train set. The data is from three main domains: 81% from Bulgarian newspapers, 16% from fiction texts, and 3% from administrative documents.

The texts of the dataset are split into sentences, and every sentence is tokenized. Then, every token is annotated with its lemma, part of speech tag, list of morphological features, head of the dependency, type of dependency relation, and other additional characteristics. The data is publicly available[2] in CoNNL-U format.[3]

3.2. BulNet

WordNet [11] is a lexical database organized on the basis of semantic features. Its development began in 1978 [21] for the English language. Currently, there are versions for over 200 languages. It is organized around the idea of synsets – cognitive synonym sets representing sets of words of the same part of speech that can be used interchangeably in a certain context. Each synset has its own short description, often referred to in the literature as a gloss, as well as a list of short examples, illustrating the exact use of the given meaning in a sentence. The Bulgarian version of WordNet is called BulNet [9].

Current BulNet consists of 22092 nouns, 9043 verbs, 8969 adjectives, and 1692 adverbs, which makes it several times smaller in volume than the English version.

When synonymous sets are separated from each other by extremely small and difficult-to-notice marks, the task becomes too complicated, even for humans. There are situations where different annotators would choose different meanings and an agreement of about 80% is achieved, which is perceived as target accuracy for a word sense disambiguation system [3].

3.3. fastText vectors

The pretrained word vectors that we are using are fastText vectors [4] for the Bulgarian language[4]. These vectors are of dimension 300 and are trained on Bulgar-

[2] https://github.com/UniversalDependencies/UD_Bulgarian-BTB
[3] https://universaldependencies.org/format.html
[4] https://fasttext.cc/docs/en/crawl-vectors.html

ian Wikipedia. FastText vectors showed the best performance among other architectures for pretrained vectors that we experimented with. One of the main reasons is that fastText works on a character level, generating n-grams of symbols on which the algorithm is trained. This makes it very well suited for morphologically rich languages, such as the Bulgarian language, where a single word can have many forms, not all of which are present in the training corpus. With the n-grams mechanism, FastText is also able to manage successfully out-of-vocabulary cases, which occur often when the training data is limited.

4. Pipeline implementation

In spaCy, there are two types of components - trainable and non-trainable. Trainable components rely on training data and machine learning algorithms. Non-trainable components rely on predefined sets of rules, by which they process the data. In our pipeline, the rule-based components are the Tokenizer and the Sentencizer. The remaining components rely on training data and machine learning modules. In the following subsections, we will discuss in more detail the algorithms which each one of the components is using. Figure 1 presents the sequence of the steps in the pipeline.

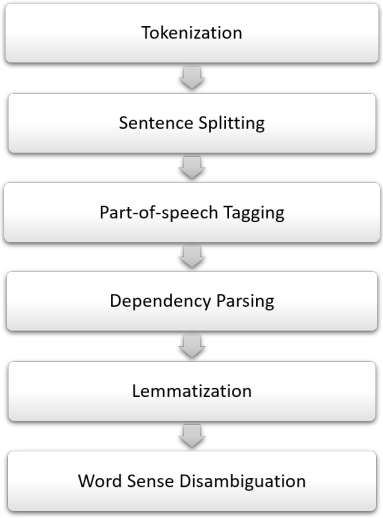

Figure 1. Sequence of the steps of the developed Bulgarian language pipeline

4.1. Rule-based components

4.1.1. Tokenizer

The tokenization is the first step of the pipeline, as the following components need input data in the form of a sequence of tokens. For the sake of modelling cor-

rectly the Bulgarian language, we created our own custom tokenizer. The Bulgarian tokenizer consists of:

- Lists of special cases, such as metrics, abbreviations, and titles,
- List of stop words,
- Regular expressions for handling tokens with special symbols, like hyphens, apostrophes,
- Regular expressions for handling punctuation.

Tokenizer exceptions consist of the following types:

- Units of measure: can be with ("см.") or without dot ("см"),
- Abbreviations - some like "изд." (from "издателство", *publishing house*) can be only in the middle of a sentence, whereas others like "др." (from "други", *others*) can be both in the middle and in the end,
- Hyphenated abbreviations, like "г-жа" (*Mrs.*),
- Capitalized abbreviations, like "БДС" (from "Български държавен стандарт", *Bulgarian State Standard*).

4.1.2. Sentence splitter

Ideally, the dependency parser algorithm should be able to learn to split sentences automatically. Unfortunately, as our training data is already split into sentences, that was not possible, and a separate rule-based algorithm had to be developed.

The sentence splitter consists of rules for treating punctuation and a variety of edge cases, connected to the uses of initials and abbreviations. Initials, such as "А.", are marked as invalid end of sentence.

This custom-built module takes as input the tokenized text. In order to split the sentences correctly, the algorithm assumes that a token can be the beginning of a sentence if:

- Starts with an uppercase letter, and
- The preceding token is not an invalid end, and
- The preceding token is end-of-sentence punctuation, or it's not one of the special cases, or is of the special cases, but is a possible end of the sentence.

In this manner, the Sentencizer is able to avoid splitting sentences where the dot is used in abbreviations, such as:

Св. Николай Чудотворец е роден 15 март 270 г. в Патара, Ликия.
(St. Nicholas the Wonderworker was born on March 15, 270 in Patara, Lycia.)

4.2. Built-in trainable components

4.2.1. POS-tagger and morphologizer

The part of speech tagger and morphologizer components are implemented by the spaCy's tagger model, which uses a linear layer with softmax activation to predict tag scores for every token's vector. The POS tagging module uses as features the token vectors, as well as information from the morphologizer, which is a trainable component that predicts morphological features and fine-grained POS tags following the Universal Dependencies UPOS[5] and FEATS[6] annotation guidelines.

4.2.2. Dependency parser

A dependency parser (DEP) is a model which analyzes the grammatical structure of a sentence. The dependency parser marks the relationships between "head" words and words that modify those heads. The spaCy parser uses a modification of the non-monotonic arc-eager transition system [7], which jointly learns dependency parsing and labelled dependency parsing. The algorithm uses a pseudo-projective dependency transformation [13], which allows it to work with non-projective trees, which may occur in languages with free word order, such as the Bulgarian language. In our training data, 279 out of 8836 sentences (3%) have non-projective dependency trees.

4.2.3. Lemmatizer

In the Bulgarian language, the lemma of a certain word cannot be determined by applying a short list of rules. One approach to the problem is to use large lists of words in all their possible forms and base form to determine the result of lemmatization. A disadvantage of the approach is that when the desired word does not appear in the list, there is no processing option. Then either the current form of the input word should be returned or an empty character string should be returned. If the dictionary of presented word forms is not comprehensive enough, both approaches will result in token mishandling and, therefore, in low recall values of the overall system.

To address this limitation, we apply a method proposed by Müller et al. [12], according to which an "EditTreeLemmatizer" is built. The method is available for use as a standalone component in the spaCy tool. In the Neural edit-tree lemmatization algorithm, the lemmatization task is treated as a classification problem. The classes represent all learned edit trees, and the Softmax function is used for computing the probability distribution over all trees for a particular token. Then the algorithm tries to apply the most probable tree and, if this is not possible, continues with the next most probable tree. An edit tree consists of the following types of nodes: inferior nodes, which split the string into a prefix, an infix, and a suffix, and leaf nodes, which apply the learned transformation. An example edit tree is shown in Figure 2.

[5] https://universaldependencies.org/u/pos/index.html
[6] https://universaldependencies.org/format.html#morphological-annotation

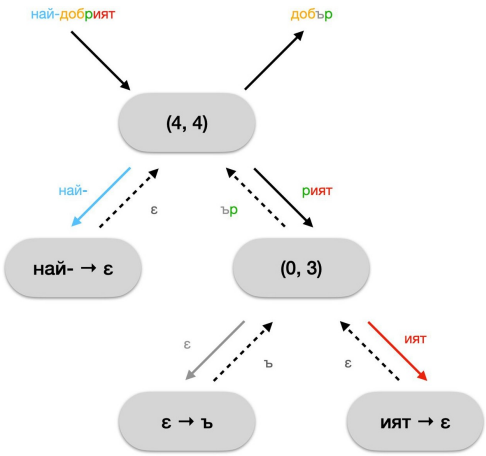

Figure 2. Edit tree for the inflected form "най-добрият" (*the best*) and its lemma "добър" (*good*)

4.3. Additional components

4.3.1. Word sense disambiguation

Word-sense disambiguation (WSD) is the task of determining the correct sense of a word in a given context [8]. It can be a supervised, unsupervised, or hybrid task. Supervised approaches utilize a corpus of sentences in which individual words are manually labelled with senses from a lexical resource, such as WordNet.

In the current pipeline, word sense disambiguation is regarded as a supervised machine-learning task. Bulgarian WordNet provides a valuable source of data, including possible senses, hypernyms, and usage examples. For each potential sense, a sample text is constructed by combining the glosses for the sense and its related hypernyms. Subsequently, these texts are passed through spaCy as distinct documents. This process results in vectors of each token in the text. Next, the vectors corresponding to each document are averaged, and each sentence is assigned a single vector. The rationale behind this approach is that different sense descriptions will have distinct vector representations, and by averaging them, the representation in the latent space will contain sufficient information for disambiguation. The final stage entails identifying the closest candidate to the target sentence in the latent space, which is accomplished by employing cosine distance for measuring distance and determining the closest candidate.

5. Experiments

5.1. Word embeddings

For the performance of the word sense disambiguation model, which uses information from the preceding steps of the pipeline, we experimented with different pre-

trained word embeddings architectures, such as BERT [5], RoBERTa [10], Flair [2] and fastText [4]. Bulgarian BERT[7] is trained on the Bulgarian version of Wikipedia and the OSCAR 2019 corpus [15], composed of 1,268,114,977 words with a total size of 14 GB, as well as with data from the online library `Chitanka.info`[8]. Bulgarian RoBERTa[9] is trained on the corpora Wikipedia, OSCAR 2019, and Newscrawl one million sentences[10].

An interesting observation is that for our task BERT and RoBERTa models have similar performance. This is most likely due to the volume of data on which the Bulgarian models were trained. For the original English models, BERT was trained on 16 GB of data and RoBERTa – on 160 GB. Such volumes of data are not available for the Bulgarian language, and accordingly the models have a comparable performance.

In our experiments, the best results were obtained with the fastText vectors, which we included in the final version of the system. A comparison of the results can be found in Table 1.

Table 1. Performance comparison of different approaches for solving the WSD task

Algorithm	Accuracy
Cosine similarity with fastText	65.24
Cosine similarity with Flair	63.99
Cos similarity with RoBERTa	62.82
PageRank – undirected graph	58.76
PageRank – directed graph	61.33

5.2. Word sense disambiguation algorithms

For the word sense disambiguation task, we experimented with two main types of algorithms – graph-based and similarity-based. One of the tested methods for resolving ambiguity involved using graph-based algorithms. This family of algorithms extracts all possible senses for a given text and builds graph representation of the text utilizing the available relations in WordNet or other similar knowledge bases. Those approaches rely heavily on the PageRank algorithm, with different modifications tailored to suit the specific problem at hand. In essence, all approaches assign weights to the nodes in the graph, which determine the most probable senses for the target text. According to Agirre et al. [1], such methods need longer texts of no less than 20 words to build a graph with weights that will ultimately converge to a single probable sense.

A crucial factor influencing the system's performance is the number of connections and senses available in the knowledge base. In the case of Bulgarian language WordNet, the available connections are fewer and lack utility for graph construction,

[7] https://huggingface.co/rmihaylov/bert-base-bg
[8] https://chitanka.info/
[9] https://huggingface.co/iarfmoose/roberta-base-bulgarian
[10] https://wortschatz.uni-leipzig.de/

since the sentence graphs may not be connected and can be composed of several disjoint graphs. As for the similarity-based approaches, the number of relations in WordNet is not significantly important because in those methods we rely heavily on the semantic relationships that have been learned from unstructured texts while training the word embeddings. Building word embeddings does not require the meticulously crafted relations encapsulated in WordNet and the embeddings' position in latent space can be used to calculate the similarity between words and sentences. In our case, the cosine similarity was used to determine the distance between different sentences. In order to deal with different sentence lengths, an average pooling was used that averages the weights of the words within the provided excerpt, resulting in an effective assignment of a single word embedding for each sentence.

Our experiments indicated that vector-based approaches prove more effective for the Bulgarian language. Table 1 summarizes the results of the different word disambiguation techniques tested.

6. Results and evaluation

6.1. Automatic evaluation

We evaluate our pipeline, based on the following metrics: TOK – tokenization accuracy, POS – part-of-speech accuracy, UAS – unlabeled attachment score for dependency parsing, LAS – labelled attachment score, LEMMA – lemmatization accuracy, WSD – word sense disambiguation accuracy, and SENT F – sentence splitting F-score. The sentence splitting score, however, is biased, as the majority of the test examples are already split into sentences.

Table 2 presents the results of the current pipeline, compared to the previously reported results [18]. We observe that our pipeline archives better results on all the included subtasks.

The previous works do not report automatic evaluation for the lemmatization and word disambiguation task, and we could compare the obtained results. However, manual error analysis, presented in the next section, shows that those two modules also show good performance for the Bulgarian language.

Table 2. Comparison of the results on the BulTreeBank dataset of the current pipeline and the previous implementation [18]

Metric	Pipeline-2020	Pipeline-2023
TOK	—	99.97
POS	94.49	98.12
LEMMA	—	93.88
UAS	89.71	89.95
LAS	83.95	84.77
WSD	—	65.24
SENT F	—	94.65

6.2. Error analysis

6.2.1. Lemmatization

A frequently occurring problem are the differences in the spelling of the base forms of the words. In BulTree Bank, the data on which the lemmatizer was trained differs from the way it is written in WordNet. Such examples are "запазя-(се)" – "запазя" (*to save oneself*), "смея се" – "смея" (*to laugh*) and others. During the system workflow, the suffixes "-(се)" and "-се", which are marking reflexive verbs, are removed before the search is executed. Unfortunately, this does not help in cases where the proposed base form is wrong. Such a case is "призовам-(се)" instead of "призовавам" (*to call out*).

Alignment of the resources would lead to an improvement of the results and an increase of the recall measure and, accordingly, of the F1 result of the overall system. On the BulTree Bank validation set an accuracy of 0.9388 is reached for the lemmatization task, which is quite a good result for a morphologically rich language such as Bulgarian. Despite the high score on this set, discrepancies with WordNet undoubtedly contribute to low recall values.

We examined closely the outputs of the edit tree lemmatization algorithm. We were able to identify three main types of mistakes made by the algorithm:

1. Errors caused by the suffixes "-(се)" and "-се".
2. Incorrect suggestion for a base form of an existing word, such as "булеварда" instead of "булевард" and "пейзажа" instead of "пейзаж".
3. Prediction non-existent words, such as "лип" instead of "липа", and "щайг" instead of "щайга".

As we can see, most of the errors come from confusion between words in the masculine gender, ending with the determination ending *-а*, and words in the feminine gender, whose base form ends with a gender ending *-а*. It is possible that with more training examples the algorithm for tree selection will be able to discern more details and choose the appropriate tree more frequently. In its current version, the model is trained on 8,907 sentences from the BulTree Bank, which is insufficient for a morphologically rich language such as Bulgarian.

6.3. Word sense disambiguation

There are two main sources of errors of the word sense disambiguation module:

1. Synsets, which are presented in BulNet with multi-token expressions, such as "черен чай" (*black tea*) and "маслодайни рози" (*oil roses*).
2. Overlapping senses – often there are cases where the meaning of a word in a particular sentence can fall in more than one of the predefined senses, and the predicted and original senses are different, while equally true. An example of such a case is shown in Table 3.

Table 3. Example of an error by the word sense disambiguation module for the meaning of the word "пристъпвам" (*to step*), where the predicted and the original sense are very close

Example	Expected sense	Predicted sense
Пристъпих напред и вдигнах ръка. (I stepped forward and raised my hand.)	*btbwn-038000141-v* Движа се като правя стъпка след стъпка в равномерен ритъм. (I move by taking step after step in a steady rhythm.)	*btbwn-038000146-v* Правя, направям една или няколко стъпки, обикновено в посоката, към която гледам, към която съм обърнат. (I take one or more steps, usually in the direction I'm looking or, facing.)

7. Conclusion

We presented the implementation of an open-source pipeline for processing the Bulgarian language, built on the natural language processing library spaCy, which shows significant results on numerous tasks. We systemized the available lists of tokenizer exceptions and successfully created new custom modules for sentence splitting and word sense disambiguation and a new neural-based module for lemmatization. Finally, we released an open-source version of the pipeline.

The presented pipeline can be used in multiple ways, some of which include: in sentiment analysis and hate speech detection tasks, by lemmatizing text and searching in a list of predefined signaling words; in machine translation, to find the right meaning of an ambiguous word and produce the right translation; in text categorization tasks, by providing additional information about the text, such as additional features of the words and sentences of its contents. These and any other applications can be built by appending the pipeline with additional components (such as one for text categorization) or integrating it with other systems.

7.1. Limitations

There are several ways in which the work on the pipeline can be improved.

First, the sentencizer can be further developed to process nested sentences, typical for the press and literature. For this, additional linguistic knowledge will be needed in order to model more complicated cases.

The lemmatization module can benefit from additional data processing, and also from more data, as a significant number of edit trees modelling the lemmatisation process of Bulgarian words can exist.

Currently, our method for word sense disambiguation is unable to process word bigrams and trigrams. When the target phrase consists of more than one word, the overall model fails because the search is only performed on a single token. To improve the word sense disambiguation module performance, it is necessary to implement a comprehensive system to deal with bigrams and trigrams. Their presence in WordNet makes searching difficult in situations where there is no complete match and searching by lemma form does not lead to successful detection. Additionally,

further preprocessing is needed to solve the problem of the difference in the word forms used in the two datasets – BulTreeBank and WordNet.

7.2. Ethics Statement

This work aims at an equal and fair distribution of language technologies for different populations, especially speakers of low-resource languages, such as Bulgarian. The pipeline will be publicly available for use and modification.

We consider that there are no potential harms for different groups, if the described tool is misused. On the contrary, such a technology can help in analyzing and filtering propaganda, misinformation and hate speech in texts.

Acknowledgements

An initial version of the sentencizer and the lists of the tokenizer exceptions were compiled by Luboslav Krastev and Daniel Traikov from Apis Europe, with the assistance of Prof. Dr. Kiril Simov from the Institute of Information and Communication Technologies of the Bulgarian Academy of Sciences. These resources, as well as the data from the Bulgarian WordNet, are provided with the kind assistance of Prof. Dr. Kiril Simov.

References

[1] E. Agirre, O. L. de Lacalle and A. Soroa, Random walks for knowledge-based word sense disambiguation, Comput. Linguist. 40(1) (2014) 57–84, https://aclanthology.org/J14-1003.

[2] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter and R. Vollgraf, FLAIR: An easy-to-use framework for state-of-the-art NLP, in: Proc. 2019 Conf. of the North American Chapter of the Association for Computational Linguistics (Demonstrations), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, 54–59, https://aclanthology.org/N19-4010.

[3] M. Bevilacqua and R. Navigli, Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information, in: Proc. 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, 2854–2864, https://aclanthology.org/2020.acl-main.255.

[4] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146, https://arxiv.org/abs/1607.04606.

[5] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proc. 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, 4171–4186, https://aclanthology.org/N19-1423.

[6] G. Georgiev, V. Zhikov, A. Ontotext, P. Osenova, K. Simov and P. Nakov, Feature-rich part-of-speech tagging for morphologically complex languages: Application to Bulgarian, EACL 2012 (2012) 492, https://arxiv.org/abs/1911.11503.

[7] M. Honnibal and M. Johnson, An improved non-monotonic transition system for dependency parsing, in: Proc. 2015 Conf. on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, 1373–1378, https://aclanthology.org/D15-1162.

[8] D. Jurafsky and J. Martin, Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, Prentice Hall Ser. in Artificial Intelligence, Pearson Prentice Hall, 2009, https://books.google.bg/books?id=fZmj5UNK8AQC.

[9] S. Koeva, S. Mihov and T. Tinchev, Bulgarian wordnet – structure and validation, Romanian J. Inf. Sci. Technol. 7(1–2) (2004) 61–78, http://lml.bas.bg/~stoyan/wordnet_logic.pdf.

[10] Y. Liu, M. Ott, N. Goyal et al., Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019), https://arxiv.org/abs/1907.11692.

[11] G. A. Miller, Wordnet: A lexical database for English, Commun. ACM 38 (1995) 39–41, https://doi.org/10.1145/219717.219748.

[12] T. Müller, R. Cotterell, A. Fraser and H. Schütze, Joint lemmatization and morphological tagging with lemming, in: Proc. 2015 Conf. on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, 2268–2274, https://aclanthology.org/D15-1272.

[13] J. Nivre and J. Nilsson, Pseudo-projective dependency parsing, in: Proc. 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Association for Computational Linguistics, Ann Arbor, Michigan, 2005, 99–106, https://aclanthology.org/P05-1013.

[14] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov and E. Marsi, Maltparser: A language-independent system for data-driven dependency parsing, Nat. Lang. Eng. 13 (2007) 95–135, http://lrec-conf.org/proceedings/lrec2006/pdf/162_pdf.pdf.

[15] P. J. Ortiz Suárez, L. Romary and B. Sagot, A monolingual approach to contextualized word embeddings for mid-resource languages, in: Proc. 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, 1703–1714, https://aclanthology.org/2020.acl-main.156.

[16] P. Osenova and K. Simov, Universalizing BulTreeBank: a linguistic tale about glocalization, in: The 5th Workshop on Balto-Slavic Natural Language Processing, INCOMA Ltd. Shoumen, Bulgaria, Hissar, Bulgaria, 2015, 81–89, https://aclanthology.org/W15-5313.

[17] E. Partalidou, E. Spyromitros-Xioufis, S. Doropoulos, S. Vologiannidis and K. Diamantaras, Design and implementation of an open source Greek POS tagger and Entity Recognizer using spaCy, WI '19: IEEE/WIC/ACM Int. Conf. on Web Intelligence, 2019, 337–341, https://doi.org/10.1145/3350546.3352543.

[18] A. Popov, P. Osenova and K. Simov, Implementing an end-to-end treebank-informed pipeline for Bulgarian, in: Proc. 19th Int. Workshop on Treebanks and Linguistic Theories, Association for Computational Linguistics, Düsseldorf, Germany, 2020, 162–167, https://aclanthology.org/2020.tlt-1.14.

[19] A. Savkov, L. Laskova, S. Kancheva, P. Osenova and K. Simov, Linguistic processing pipeline for bulgarian, in: Proc. 8ht Int. Conf. on Language Resources and Evaluation (LREC'12), 2012, 2959–2964, http://www.lrec-conf.org/proceedings/lrec2012/pdf/829_Paper.pdf.

[20] K. Simov, Z. Peev, M. Kouylekov, A. Simov, M. Dimitrov and A. Kiryakov, Clark-an xml-based system for corpora development, in: Proc. of the Corpus Linguistics

2001 Conference, 2001, 558–560, http://bultreebank.org/wp-content/uploads/2017/04/BTB-TR06.pdf.

[21] P. Vossen, Wordnet, eurowordnet and global wordnet, Revue française de linguistique appliquée 7 (2002) 27–38, https://doi.org/10.3917/rfla.071.0027.

Received on March 30, 2023
Accepted on May 7, 2023

Melania Berbatova and Filip Ivanov

Faculty of Mathematics and Informatics
Sofia University "St. Kliment Ohridski"
5 James Bourchier Blvd.
1164 Sofia
BULGARIA

E-mails: msberbatov@fmi.uni-sofia.bg
phil.ivanov@outlook.com