Part of Speech Tagging for Serbian language using Natural Language Toolkit

Boro Milovanović, Ranka Stanković

***Abstract*—While complex algorithms for NLP (Natural language processing) are being developed, base tasks such as tagging remain very important and still challenging. NLTK (Natural Language Toolkit) is a powerful Python library for developing programs based on NLP. We try to leverage this library to create a PoS (Part of Speech) tagger for a contemporary Serbian language. Eleven different models were created by using NLTK tagging API. The best models are transformed with the Brill tagger to improve the accuracy. We trained the models on the tagged dataset counting 180,000 tokens. The best results on the test set of 20,000 tokens were demonstrated with the Perceptron tagger: 92,52 – 95,76% accuracy for the different tagsets.**

***Index Terms*—Natural Language Processing; Machine Learning; Neural Network.**

I. Introduction

In the last couple of years, a big advancement in the field of Natural Language Processing has occurred. There are state-of-the-art language models that perform exceptionally in various language tasks [1-3]. The applications are getting broader, the algorithms are more complex [4]. Beneath the surface, there is a limited set of the tasks that still pose challenges to the researchers. Small improvements in the basic tasks pose immediate benefits to the tasks which are performed later in the pipeline.

One basic task is PoS (Part of Speech) tagging, a process of assigning a part of speech category to each token in the text. The program that performs tagging is called tagger. The taggers can be created in multiple ways. In this paper, we will create a tagger for Serbian with a help of a Python library NLTK (Natural Language Toolkit). Besides just exposing more than 50 corpora and lexical resources, NLTK is used for making programs that handle human language data, ranging from tokenization to semantic reasoning. NLTK API makes it possible to create multiple standalone tagger models as well as to combine them. We are going to try and test every model available in the version 3.5 released in March 2020. Having a plethora of different algorithms makes this library a good choice for a research.

Serbian language belongs to a group of low-resource languages so there's a modest research on this topic. First attempts to create an automatic PoS tagger for Serbian relied on a dictionary. Delić et al. used custom transformations and rules [5]. Utvić created a parameter file TT11 for a TreeTagger [6]. Later attempts relied on CRF (Conditional Random Fields) [7-8] which is among supported technologies by NLTK and will be used for training one of the taggers.

Introducing the dataset and the tagset will be done in the Section 2. The creation of multiple taggers is presented in Section 3. The results will be shown in Section 4 and briefly discussed in Section 5. We will conclude with Section 6.

II. Resources

An automated tagger is created by training on an annotated dataset. These resources are extremely valuable because they are expensive to produce. Dataset used in this paper is composed of different annotated text collections (Table I). All texts are either originally written in Serbian or translated to it. *1984* is a novel by George Orwell, part of MULTEXT-East resources [9]. *INTERA* (Integrated European language data Repository Area) is a project that produced multilingual corpus on law, health and education [10]. *Around the world in 80 days* is a novel by Jules Verne annotated during SEE-ERA.net project [11]. *ELTeC* (European Literary Text Collection) is a multilingual collection of the novels written between 1840 and 1920 [12]. *ELTeC-srp* is the Serbian part of the *ELTeC*. *History*, *Floods* and *Švejk* are three short collections originated during the same research [7]. *History* is made of several chapters from a History textbook for elementary schools. *Floods* is a newspaper collection reporting on floods in Serbia in 2014. *Švejk* is an excerpt from a novel *The Good Soldier Schweik* by Jaroslav Hašek.

TABLE I
Dataset structure

Collection	Tokens	Words
1984	108,133	69,706
INTERA	65,767	55,725
Around the world in 80 days	7,382	5,970
History	5,277	4,230
ELTeC-srp	5,118	4,236
Floods	4,671	3,813
Švejk	3,298	2,678

In total there are 199,646 tokens. Among them, 31,139 tokens are unique. An example of tagged tokens is given in the

Boro Milovanović is a PhD student attending Intelligent Systems, an interdisciplinary program at the University of Belgrade, Studentski trg 1, 11000 Belgrade, Serbia (e-mail: bmilovanovic@tesla.rcub.bg.ac.rs).

Ranka Stanković is with the Faculty of Mining and Geology, University of Belgrade, Djušina 7, 11000 Belgrade, Serbia (e-mail: ranka.stankovic@rgf.bg.ac.rs).

Table II. Every row contains 5 tab-separated values, that are described below.

TABLE II
DATASET EXAMPLE ROWS

SentenceId	Token	N_POS	UD_POS	Lemma
5	velikog	A:am	ADJ	velik
5	doba	N:n	NOUN	doba
5	.	SENT	PUNCT	.

There are 10,890 sentences in the data set, labeled by the *SentenceId*, a first value in the row. Actual word that is tagged is contained in the *Token* column. Its respective *Lemma* is in the last column. There are two PoS tags for every token, originated from two different tagsets. Tagset is a collection of tags. UD_POS is a Universal Dependency tagset [13]. N_POS is a tagset used in Serbian Morphology Dictionary [14] expanded with a gender category. From the given data we extracted token, N_POS and UD_POS tag. We stripped gender from the N_POS and got a third tagset which we called SMD_POS. All three tagsets are used in a further research.

There are 31139 unique tokens in the dataset and 17 UD PoS categories: adjective (ADJ), adposition (ADP), adverb (ADV), auxiliary verb (AUX), coordinating conjunction (CCONJ), determiner (DET), interjection (INTJ), noun (N), numeral (NUM), particle (PART), pronoun (PRON), proper noun (PROPN), punctuation (PUNCT), subordinating conjunction (SCONJ), symbol (SYM), verb (VERB) and other (X). The tags are distributed as shown in the Table III.

TABLE III
PART-OF-SPEECH CATEGORY DISTRIBUTION AND MAPPING

UD_POS	COUNT	%	N_POS
NOUN	42936	21.51%	N, N:m, N:f, N:n
PUNCT	31477	15.77%	PUNCT, SENT
VERB	20599	10.32%	V, V:m, V:f, V:n
ADJ	18949	9.49%	A, A:am, A:an, A:af
ADP	16540	8.28%	PREP
AUX	13592	6.81%	V, V:m, V:f, V:n
CCONJ	9374	4.70%	CONJ
ADV	8998	4.51%	ADV
DET	8599	4.31%	PRO, PRO:m, PRO:f, PRO:n
PART	7976	4.00%	PAR
SCONJ	6304	3.16%	CONJ
PRON	5751	2.88%	PRO, PRO:m, PRO:f, PRO:n
PROPN	3949	1.98%	N, N:m, N:f, N:n
NUM	3634	1.82%	NUM, NUM:m, NUM:f, NUM:n
X	858	0.43%	X, PREF
INTJ	110	0.06%	INT

Table III also shows mapping between UD_POS and N_POS tags. In most cases it is one-to-one relation but there are some differences between tagsets. N_POS doesn't differentiate between VERB and AUX, CCONJ and SCONJ, NOUN and PROPN. On the other hand, it separates SENT from PUNCT and PREF from X.

The most frequent word category is Noun, followed by a Punctuation and a Verb. Cumulative distribution of PoS categories is shown in Figure 1. First five categories account to 57% of all tokens. These numbers help us in creating the taggers and interpreting their performance.

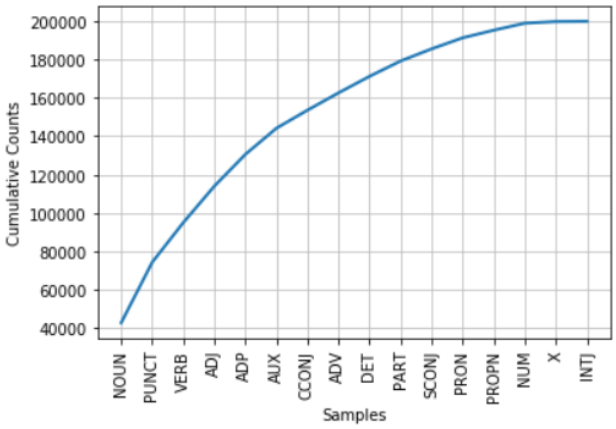

Fig. 1. Word PoS category cumulative distribution

For a complete evaluation of the taggers, we need a data from a different domain and origin. It will be exempted from the training and validation phases and will be used at the end of the evaluation phase to test generalization potential of the created models. We took this data from a *ParCoTrain* dataset [15] and mapped their tagset (which we called PCT_POS) to the SMD_POS tagset, which is shown in Table IV. This is an excerpt from a novel *Enciklopedija Mrtvih* [16], having 23,886 tokens in 946 sentences.

III. TAGGING

After the resources are ready, the process of tagging is made simple with the help of NLTK. There are a plenty of tagger models packaged in NLTK that can be trained. Every tagger has an evaluation procedure that strips down the tags from the given text, tags the text with the newly created tagger and reports the accuracy on all tokens. This measure will be used for comparing different taggers.

The simplest model in NLTK is *Default* tagger which tags every token with one selected PoS category [17]. Because the noun is by far the most represented PoS tag, the accuracy for this model on all tokens will be exactly 21,51%. This is the baseline tagger, point of reference for all other tagger models.

By applying more rules regarding the token format, a *RegExp* tagger is produced. It is initialized with the list of regex rules which are executed in the defined order. If one pattern doesn't match the given token, a next one is picked. At the end of the chain, there is a rule which states that the word is of Noun category – same as for *Default* tagger. Obviously, this model

will perform better than the baseline tagger, but making the right rules demands significant effort. We did not invest much time in producing the rules, because we had not seen significant improvements after adding new regular expressions.

TABLE IV
MAPPING BETWEEN SMD_POS AND PCT_POS TAGSETS

SMD_POS	PCT_POS
N	NOM:com, NOM: col, NOM:nam, NOM:num, NOM:approx
V	VER, VER:aux
PRO	PRO:per, PRO:intr, PRO:dem, PRO:ind, PRO:pos, PRO:rel, PRO:ref
NUM	NUM, NUM:car, NUM:ord, NUM:cal, PRO:num
A	ADJ:comp, ADJ:sup, ADJ:intr, ADJ:dem, ADJ:ind, ADJ:pos, ADJ:rel
ADV	ADV:comp, ADV:sup, ADV:intr, ADV:rel, ADV:ind
CONJ	CONJ:coor, CONJ:sub
PREP	PREP
PAR	PAR
INT	INT
SENT	SENT
PUNCT	PONC, PONC:cit
X	STR, ABR, LET, PAGE, ID

Affix tagger takes only prefixes or suffixes of fixed length in consideration. It learns the most frequent tag in the dataset for a given affix. If the word is shorter than 5 characters, tagger returns tag "None".

The most frequent PoS category is Noun but there are plenty of other words with high frequency that are not nouns. This is the idea for the *Lookup* tagger (implemented through *UnigramTagger* class). It remembers the tags for the most frequent words, while marking all the other words as None. A number of the most frequent words for which the tags should be stored is configurable. Figure 2 shows how accuracy of the model measured on the whole dataset improves with the number of the stored tags.

It is also seen that the best accuracy is achieved when the tags are remembered for all the words. This is how *Unigram* tagger behaves. It stores the most frequent PoS tag for all the tokens and marks every appearance of that token with it.

The taggers mentioned above determined the tags solely on the given token. *N-gram* tagger takes the context of the token in consideration. *Bigram* tagger takes that token and the one preceding it. *Trigram* tagger takes the two tokens before the observed one. *N-gram* taggers are most effective when combined with lower-level models.

While constructing the sequential tagger, it is possible to define a back off tagger which will take over when the current tagger is not able to determine a tag for a token (returning tag None). This is the back off chain that we tested: *Trigram – Bigram – Affix – Unigram – Default.*

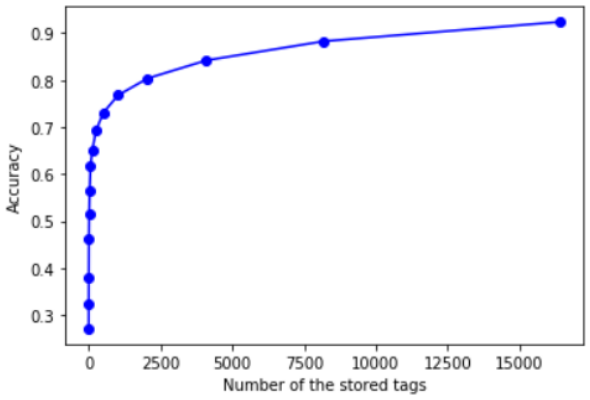

Fig. 2. Dependence of the Lookup tagger accuracy on the number of the stored tags

Aside embedded sequential tagger models available in the *nltk.tag.sequential* package, there are custom made models available in separate subpackages. *CRF* tagger is based on Conditional Random Fields [18]. *HMM* tagger is based on Hidden Markov Models [19]. Training on an averaged, one-layer neural network produces *Perceptron* tagger [20]. *TnT* tagger is short of Trigrams'n'Tags and it uses a second order Markov model to produce tags for an input sequence [21]. We declared the Unigram tagger as a back off tagger because *TnT* does not automatically work with unseen words.

Any of the mentioned taggers can be improved with the help of *Brill* tagger [22]. It uses a configurable set of rules to correct the errors and improve the total accuracy. Best performance is shown by a *brill24* set of rules. We apply *Brill* to the top performing models to try to increase the accuracy.

IV. RESULTS

All tagger models except *Default* and *RegExp* tagger are created by training on an annotated dataset. We were training on the 90% size of the original dataset size and measured accuracy on a remaining 10% size with 10-fold evaluation[1]. Results of the trained tagger models can be seen in Table V, with the accuracy calculated as

$$Accuracy = \frac{Number\ of\ correctly\ predicted\ tags}{Number\ of\ all\ tags} \quad (1)$$

The taggers are trained on an *Intel® Core™ i7-8750H CPU @ 2.20Ghz* with 16 GB RAM. It can be seen from the table that training all taggers for N_POS tagset took much more time due to the added gender information. When the processor had a task in parallel, other than training, the training times were twice as high.

Accuracy scores, for all taggers, are very close between the UD_POS and SMD_POS tagsets. This is expected because the number of the tag categories and their distribution is similar.

[1] Code for training and evaluation, example dataset and results are available at: https://github.com/bmilovanovic/pos-tagging-serbian.

However, taggers performed remarkably worse for the N_POS tagset. Information about the gender added complexity so simpler taggers could not deal with it easily. However, the best tagger models, CRF and Perceptron, kept the accuracy over 90 percent even with the gender information. We took these two taggers to the additional evaluation.

TABLE V
ACCURACY OF TAGGERS FOR EACH TAGSET

Tagger	UD_POS	SMD_POS	N_POS
Default	21.50	23.50	12.15
RegExp	23.20	25.33	13.06
Affix	88.34	87.12	81.64
Lookup	43.26	40.87	40.70
Unigram	90.56	88.79	84.87
Bigram	91.56	90.17	86.58
Trigram	91.50	90.01	86.38
CRF	**93.77**	**93.72**	**90.16**
HMM	44.28	49.80	45.58
Perceptron	**95.61**	**95.76**	**92.52**
TnT	90.83	90.51	86.95
Training Time	1143s	1343s	3074s

Useful tagger model is one which generalizes well to the text from the other domains. That's why we tested our best taggers on the text that stayed out of the training and validation phases. Results can be seen in Figure 3.

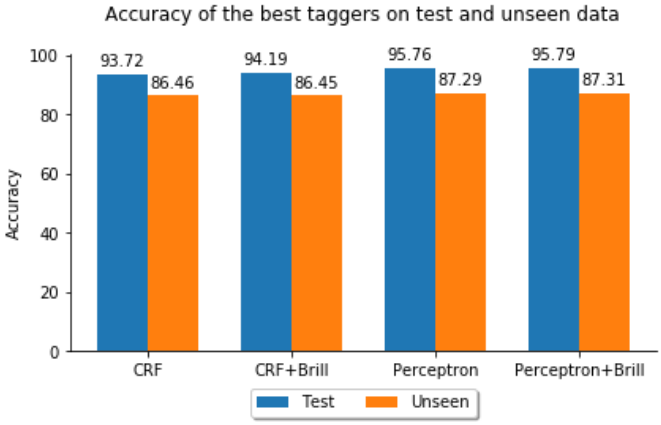

Fig. 3. Accuracy of the CRF and Perceptron variants on the test and unseen data

Taggers shown in the Figure 3 are trained on the SMD_POS tagset because it was the most like the one in unseen data so we could map between two easily. CRF and Perceptron taggers saw a small improvement with the corrections from Brill tagger. However, all four models saw a fall of about 8% in accuracy for the unseen data.

For an additional insight into the performance of the taggers, we calculate precision, recall and F1 at a tag level for the Perceptron + Brill tagger, as followed

$$Precision = \frac{T_p}{T_p + F_p} \quad (2)$$

$$Recall = \frac{T_p}{T_p + F_n} \quad (3)$$

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

The scores are calculated by iterating over all tokens, comparing the predicted tag versus actual tag and then aggregating the cases. True positive (T_p) is when these tags are identical. False positive (F_p) is the tag that is predicted but is different than actual tag. The actual tag, that is not predicted correctly, is determined as false negative (F_n). Table VI displays the results.

TABLE VI
TAG-LEVEL METRICS FOR PERCEPTRON + BRILL TAGGER EVALUATED ON UNSEEN DATA

SMD_POS	Precision	Recall	F1
N	0.91	0.99	0.92
PUNCT	0.99	0.99	0.99
V	0.91	0.93	0.92
A	0.79	0.53	0.64
PREP	0.99	0.98	0.98
CONJ	0.91	0.93	0.92
ADV	0.80	0.68	0.73
PRO	0.53	0.91	0.67
PAR	0.75	0.93	0.83
NUM	0.63	0.65	0.64
SENT	1.00	0.99	1.00
X	0.53	0.04	0.07
INTJ	0.38	0.33	0.35
Total	0.87	0.87	0.87

Many tags have low F1 scores, with X having the lowest one. Recall for X is only 0.04 which means that there are a lot of tokens with actual tag X, that the tagger did not predict as such. However, overall accuracy (87.31) is higher than most tags have because the precision is high for N and PUNCT tags which are the most frequent.

V. DISCUSSION

Looking again at the Table VI, we can notice fluctuation in performance between various tags. This is probably due to the differences in the tagging practice for the training and evaluation sets. In the process of preparing data, there are multiple tagsets and annotators. This is too many factors for an automated tagger to have the performance near maximum.

Although no research is conducted using different NLP tool and the exact same resources, there is an evidence of better

performance in PoS tagging a contemporary Serbian language [23]. Their performance on unseen data shows 0.88 precision with Spacy tagger and 0.93 with TreeTagger19 while the best tagger produced in this research achieves 0.87. The technologies in this research are not able to produce us a generalized, multi-purpose, all-around PoS tagger that can be a standard for a Serbian language.

Best performance in this research is achieved with the Perceptron tagger, a neural network which is more than a decade old. Since then, a breakthrough with deep learning has happened, so there's a strong belief that further improvements can be made with the latest neural network models [24]. However, there is a doubt if these models, because of their complexity, will ever be available in NLTK.

VI. Conclusion

We used NLTK, a Python library, to create 11 automated PoS taggers for a contemporary Serbian language. Models were trained on 180,000 tokens and evaluated on 20,000 tokens. The top performing models were improved with the help of Brill tagger and then tested on both familiar and an unfamiliar text. Best performance is shown by the Perceptron tagger: 92,52 – 95,76% accuracy for the different tagsets.

Acknowledgment

B.M. would like to thank BSc Gorana Marković for proofreading the manuscript.

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Oct. 2018

[2] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," Feb. 2020

[3] Z. Zhang, J. Yang, and H. Zhao, "Retrospective Reader for Machine Reading Comprehension," Jan. 2020

[4] B. Li, N. Jiang, J. Sham, H. Shi, and H. Fazal, "Real-world Conversational AI for Hotel Bookings," Proc. International Conference on Artificial Intelligence for Industries, Laguna Hills, California, USA, Sep. 2019

[5] V. Delić, M. Sečujski, and A. Kupusinac, "Transformation-based part-of-speech tagging for Serbian language," Proc. 8th WSEAS International Conference on Computational intelligence, man-machine systems and cybernetics, Tenerife, Spain, Dec. 2009

[6] M. Utvić, "Annotating the Corpus of Contemporary Serbian," INFOtheca, vol. 12 no. 2 pp 36a-47a, Dec. 2011

[7] M. Constant, C. Krstev, and D. Vitas "Lexical Analysis of Serbian with Conditional Random Fields and Large-Coverage Finite-State Resources", Proc. 7th Language and Technology Conference (LTC), Poznan, Poland, Nov. 2015

[8] N. Ljubešić, F. Klubička, Ž. Agić, and I. Jazbec, "New inflectional lexicons and training corpora for improved morphosyntactic annotation of Croatian and Serbian", Proc. 10th International Conference on Language Resources and Evaluation (LREC'16) pp. 4264-4270, Portorož, Slovenia, May 2016

[9] C. Krstev, D. Vitas, and T. Erjavec, "MorphoSyntactic Descriptions in MULTEXT-East | the Case of Serbian," Informatica, vol. 28 no. 4 pp. 431–436, Dec. 2004.

[10] M. Gavrilidou, P. Labropoulou, S. Piperidis, V. Giouli, N. Calzolari, M. Monachini, C. Soria, and K. Choukri, "Language Resources Production Models: the Case of the INTERA Multilingual Corpus and Terminology," Proc. Fifth International Conference on Language Resources and Evaluation (LREC'06), Genoa, Italy, May 2006

[11] D. l. Tufis, S. Koeva, T. Erjavec, M. Gavrilidou, and C. Krstev, (2009). "Building Language Resources and Translation Models for Machine Translation focused on South Slavic and Balkan Languages". Scientific results of the SEE-ERA.NET pilot joint call, pp 5, Oct. 2009

[12] Distant Reading for European Literary History, a COST Action funded by the Horizon 2020 Framework. https://www.distant-reading.net/, Mar. 2020

[13] M. d. Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning, "Universal Dependencies: A cross-linguistic typology," Proc. Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, May 2014

[14] C. Krstev and D. Vitas, "Serbian Morphological Dictionary – SMD," University of Belgrade, HLT Group and Jerteh, Lexical resource, 2.0, 2015

[15] A. Balvet, D. Stošić, and A. Miletić, (2014). TALC-Sef a Manually-revised POS-Tagged Literary Corpus in Serbian, English and French. Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), pp. 4105-4110, Reykjavik, Iceland. May 2014

[16] D. Kiš, *Enciklopedija mrtvih*, Beograd, Jugoslavija, Globus, 1983

[17] S. Bird, E. Klein, and E. Loper, "Automatic Tagging" in *Natural Language Processing with Python*, Sebastopol, California, USA: O'Reilly, 2009, ch. 5, sec. 4, pp. 198

[18] T. Peng and M. Korobov, pythoncrfsuite. https://python-crfsuite.readthedocs.io, 2014

[19] X. Huang, A. Acero, H. Hon, "Hidden Markov Models" in Spoken Language Processing, Upper Saddle River, New Jersey: USA, ch. 8, sec. 2, pp. 378-391

[20] H. Daume III, "Frustratingly Easy Domain Adaptation," Proc. 45th Annual Meeting of the Association for Computational Linguistics, Prague, Czech Republic, June 2007

[21] T. Brant, "A Statistical Part-of-Speech Tagger," Proc. Sixth Applied Natural Language Processing Conference, Seattle, Washington, USA, 2000

[22] E. Brill, "A simple rule-based part of speech tagger", Proc. Third conference on Applied natural language processing (ANLC '92), Stroudsburg, Pennsylvania, USA, Mar. 1992.

[23] R. Stanković, B. Šandrih, C. Krstev, M. Utvić, and M. Škorić, "Machine Learning and Deep Neural Network-Based Lemmatization and Morphosyntactic Tagging for Serbian," Proc. International Conference on Language Resources and Evaluation, pp. 3954-3962, May 2020

[24] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual String Embeddings for Sequence Labeling," Proc. 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA Aug. 2018