Implementation of the Serbian language POS taggers

using the NLTK library

1Nikola V. Vukotić is with University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: nikolaraca@elfak.rs

2Sofija Č. Petrović is with University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: sofija.petrovic@elfak.rs

3Suzana R. Stojković is with University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: suzana.stojkovic@elfak.ni.ac.rs

Nikola V. Vukotić1, Sofija Č. Petrović2, Suzana R. Stojković3

*Abstract –*This paper describes research on different methods for implementing part-of-speech taggers for the Serbian language. Experiments were conducted on the SrWaC corpus, which consists of 10 million tokens collected from .rs web domain. Algorithms from NLTK library were used for creating tagging models. The tagset was modified and adapted for semi-deep tagging. The experiments involved preprocessing of the corpus to correct tagging mistakes and reversing the word order in sentences to obtain a reversed corpus, which is used for creating backwards tagging model. To achieve greater accuracy, a tagger that combines n-gram taggers was implemented using the NLTK library. However, the best result achieved during tagging was with the Perceptron tagger, which reached 95% accuracy. The goal of this work is, among other things, to ensure that tagging is fast, as the intention is to use the best models for a syntactic analysis tool for the Serbian language.

*Keywords –* POS tagger, NLTK library, Serbian language POS tagger, forwards tagging, backwards tagging.

1. Introduction

Part-of-speech (POS) tagging is a process of assigning labels to words in a text that denotes their parts of speech and morphosyntactic form. POS tagging is the one of necessary first steps in many Natural Language Processing (NLP) tasks such as named entity recognition, question answering, sentiment analysis… To develop a good POS tagging tool for any natural language, a tagset and an appropriate annotated text corpus are needed.

A. Serbian language tagsets and corpora

The main characteristic of Slavic languages, including Serbian, is the existence of many morphosyntactic forms of certain types of words. Therefore, it is very difficult to define a comprehensive tagset for the Serbian language. Because of that, different available corpora use different tagsets. Some of them will be discussed below.

SrpLemKor [1] is a Serbian lemmatized and PoS annotated corpus. It doesn’t use Serbian alphabet, but it uses a special code system AURORA that replaces special characters from Serbian alphabet with specific codes [2]. N\_POS tagset is used in this corpus. It contains only 16 tags and provides information about part of speech of word, but not about its morphosyntactic form.

SrWaC [3] is a Serbian web corpus that is collected from the *.rs* top-level domain. Originally, it contains around 600 million tokens, but it is usually too big for processing. The corpus is represented with .xml file and each xml element contains a sentence, separated into tokens with specific lemmas and tags. The paper [4] describes the motive for the creation of the mentioned corpus, as well as more detailed information about it. MULTEXT-East Croatian part-of-speech tagset [5] is used in this corpus. It contains 675 different tags containing full information about parts of speech and morphosyntactic forms of the words. However, even this set of labels is not complete. For example, it does not contain information whether a pronoun is a nominal or an adjectival pronoun, which is very important for its syntactic role in the sentence.

B. Related work

The problem of creating the Serbian language tagger is already solved in the last several years. The approaches in developing Serbian tagger can be classified into two categories. The first category is the traditional approach that uses probabilistic methods to create models [6-8], while the second category is based on modern transformer language models [9-10].

PoS tagging for the Serbian language that is described in the paper [6] is implemented using NLTK Library. Dataset that is used is composed from multiple data sources. The best accuracy is accomplished using Perceptron tagger and it equals to 92.52 for N\_POS tagset, i.e. this paper discusses shallow tagging.

The paper [7] presents newly developed inflectional lexicons and manually annotated corpora of Croatian and Serbian. They introduced hrLex and srLex, two freely available inflectional lexicons of Croatian and Serbian, and described the process of building these lexicons. Furthermore, they introduced hr500k, a manually annotated corpus of Croatian, 500 thousand tokens in size. In the paper, newly developed CRF tagger showed better results than existing HunPos tagger. This tagger uses MULTEXT East Serbo-Croatian tagset that is described below and cannot be trained by another corpus.

Another implementation of the Serbian language processing tools was found in the SrbAI library. Its implementation is described in the git repository [8]. Among other things, it contains methods for recognizing types of words (PoS tagging). The HunPos model, created for the Serbian and Croatian languages, was used. It is a tagger based on the HMM algorithm, which will be described further in this paper. This tagger can be used only in Windows OS.

The paper [9] describes a transformer model pre-trained on 8 billion tokens of crawled text from the Croatian, Bosnian, Serbian and Montenegrin web domains. The transformer model is evaluated on the tasks of part-of-speech tagging, too. The BERTić model is made available for free usage and further task-specific fine-tuning through HuggingFace. BERT language model was used for tagging words in a sentence. Downside of this model is that it requires especially massive and diverse training data to stand out comparing to other transformer language models. If the training and test data were similar (like for Serbian language corpus) then evaluation results were not that good. Applications that are supposed to use this model require significant computational power to provide results quickly in real life. BERTić seem to be better for some other NLP tasks which require more world and commonsense reasoning knowledge than those with morphosyntactic tagging task.

The paper [10] presents experiments on Slovenian, Croatian and Serbian morphosyntactic annotation and lemmatization between the former state-of-the-art for these three languages and one of the best performing systems at the CoNLL 2018 shared task, the Stanford NLP neural pipeline. Their experiments confirmed that the neural approach yields significant improvements in tagging, especially because of better long-range dependency modelling and more distributional semantic information available. Besides the need for more training data and more computational power, some of the common tagging mistakes were even more frequent using neural approach. The example of it is the disambiguation between homonymous conjunctions (Cc, Cs) and adverbs (Rgp) for Croatian and Slovenian (e.g. već, tako, zato), which does come as a surprise as this distinction requires long-range information which should be more available in the neural approach.

C. Motivation for the present work and paper overview

The goal of this work is to develop a tagger which would be used in „Serbian Sentence Syntax Structure Markup Tool” [11]. The tool needs semi-deep tagging – part of speech label is not enough, but the full morphosyntactic form is not necessary. Because of that, the shallow taggers, such as those described in [6], cannot be used. The tagger described in [7] uses deep labels, but its tagset, as noted below, does not contain information whether the pronoun is a nominal or an adjectival pronoun. Failing of the tagger [8] is its inability to work in UNIX OS.

For the same reasons, the taggers [7], [9] and [10] do not have enough information about pronouns because they also use MULTEXT [5] tagset. The tagger [9] can also work with universal PoS tagset, but in that case tagging is shallow.

Therefore, it is needed to develop a new Serbian tagger. We choose to train the NLTK taggers. SrWaC corpus is used for training because it contains the most lemmas and quite a good tagset. Some corrections in the corpus must be done.

This paper is organized in the following way: Section II presents the POS tagging algorithms from NLTK library that are used in this paper. In section III, modifications done in the SrWaC corpus are described. Section IV shows evaluation metrics that are used in comparison of tested algorithms. Trained models are compared in Section V. Section VI summarizes results of the created taggers and discusses a possible future work.

1. NLTK Library

In this paper, algorithms from the NLTK library were used for labeling sentences in the Serbian language. NLTK library contains classes and interfaces for part-of-speech tagging, which can be defined as a process of classifying words into their parts of speech and labeling them accordingly. To implement the tagger, two packages from the mentioned library were used: *nltk.corpus.reader* and *nltk.tag*.

*nltk.corpus.reader* package defines a collection of corpus reader classes, that can be used to access the content of different corpora. *TaggedCorpusReader* module reads simple part-of-speech tagged corpora. Paragraphs are fragmented using blank lines and sentences and words are tokenized using the default or custom tokenizers. Words are parsed using *nltk.tag.str2tuple* and “/” is used as a default separator of word and tag (word / TAG). POS tags are normalized to upper case.

*nltk.tag* package defines multiple taggers, depending on the algorithm they are using. In this research, the following taggers were tested.

A. N-gram tagger (Unigram, Bigram and Trigram)

It is a probabilistic model trained on corpus of data. It is built by counting how often word sequences occur in corpus and then estimating probabilities [12].

B. Combined tagger

Created using Unigram, Bigram and Trigram model by chaining them. If one tagger does not know how to tag a word, it passes the word to the next backoff tagger.

C. HMM tagger

Hidden Markov Model is a stochastic technique for tagging which represents a special case of Bayesian classification. As described in the paper [13], the HMM tagger divides events, i.e. states and corresponding variables, into hidden X states and observed Y states. For example, a hidden event can be a phoneme spoken in a speech signal or one of its parameters, or a lexeme and a word label in a text input sentence (that is, a label), and the observed event is then either a label (for PoS), or a recognized word in sentences (for WSD).

D. Perceptron tagger

The perceptron tagger is a supervised machine learning algorithm that is trained on a labelled corpus of a text. The goal of the algorithm is to learn a function that maps each word in a sentence to its corresponding part-of-speech tag, based on a set of features that describe the word and its context.

During training, the perceptron algorithm updates a weight vector that is used to calculate a score for each possible tag for each word, based on the features. The algorithm then selects the tag with the highest score as the predicted tag for that word. The perceptron algorithm repeats this process for each word in the sentence, updating the weight vector after each prediction. This process is repeated for multiple epochs, or passes over the training data, to improve the accuracy of the tagger. During testing, the perceptron tagger uses the learned weight vector to predict the part-of-speech tag for each word in a new, unlabelled sentence, based on the same set of features used during training. [14]

1. Modified SrWaC corpus

Due to its original size, the corpus is modified to 10 million tokens and the modified version is represented with .txt. file. Each line contains one pair of token | label. The used tagset is modified, too. For the pronouns, the subtypes nominal and adjectival are introduced. For other word types, labels are filtered – some information irrelevant to parsing is discarded. In this way, number of labels is also reduced to 96 and used labels can be seen on Table 1. For that reason, a tagger trained to classify words using these labels cannot be called “deep”, but “semi-deep” tagger. This means that it determines type, subtype and case of a word in a sentence.

Table I

Labels used for tagging

|  |  |  |
| --- | --- | --- |
| **Type of Word** | **Subtypes** | **Case** |
| A – adjective | G – general, P – passive verb, S –possessive | N,G,D,A,V,I,L |
| N – noun | / | N,G,D,A,V,I,L |
| C – connector | C – coordinating, s subordinating | / |
| P – pronouns | N – nominal, p – adjectival pronoun | N,G,D,A,V,I,L |
| M – numbers | / | N,G,D,A,V,I,L |
| V –verb | A – auxiliary, M – main | / |
| R – adverb | R – verbal, G – general | / |
| S – preposition | / | G,D,A,I,L |
| I – interjection | / | / |
| Q – particle | R – affirmative, Z – negative, O–modal, Q – question | / |
| Y – abbreviation | / | / |
| Z – interpunction | / | / |
| X – the rest | F – foreign words | / |

The srWaC corpus is not tagged “by hand”. Therefore, there are some tagging mistakes in initial corpus that later lead to the same mistakes while using the trained model for tagging. Some of these examples are:

* 1. Wrong tag for particle “da”. It is tagged as conjunction instead of particle in original corpus.
  2. Wrong tag for adjective pronouns “ko”. It is tagged as conjunction instead of pronoun in original corpus.
  3. Wrong tag for question particle “da li”. “Da” is tagged as conjunction and “li” is tagged as particle instead of being tagged as syntagma as question particle.
  4. Wrong tag for word “sve” when it comes before adverb. It is tagged as general adjective instead of general adverb.
  5. Wrong tag for affirmative particle “da”. It is tagged as conjunction.

These examples are corrected during preprocessing phase by applying grammar rules using Python programming language (the preprocessing does not improve metric results):

1. If word “da” was found before verb, its tag was corrected to particle.
2. If word “ko” was found before verb, its tag was corrected to pronoun.
3. If words “da” and “li” were found together, their tags were corrected to question particles.
4. If word “sve” was found before verbal or general pronoun, its tag was corrected to general pronoun.
5. If word “sve” was found before an adverb, its tag was corrected to general adverb.
6. Tag for words “svi” and “sva” were corrected to adjective pronoun.
7. Tag for words “koji”,”koje”,”koja” were corrected to adjective pronoun.
8. If word “da” was found before comma, its tag was corrected to affirmative particle.
9. Evaluation metrics

All created models were evaluated in the same way. The *sklearn.metrics* module was used for the evaluation. The metrics used for the evaluation are described below:

* *Accuracy* – returns the fraction of correctly classified samples (with normalization on) or the number of correctly classified samples (with normalization off).
* *Precision* – computes the ratio in equation 1

(1)

where *tp* is the number of true positives and *fp* is the number of false positives. It takes values from interval [0,1].

* *Recall* – computes the ratio in equation 2

(2)

where *tp* is the numbers of true positives and *fn* is the number of false negatives. It takes values from interval [0,1], too.

1. Experiments and results

The evaluation of the results was done by using Unigram, Bigram, Trigram, Combined, HMM and Perceptron taggers. Previously, the corpus was processed as described in the previous chapters, since certain mistakes were observed.

Tag of the word is dependent on the context in which it is used, i.e. the tag of the current word in sentence is dependent on the tags of the previous and of the following words. Therefore, taggers can do in forwards and in the backwards orders. In forwards order, tags of the previous words in sentence are used to decide which tag to assign to the current word. In backwards order, tags of the following words in sentence are used. The best solution would be to use both orders. If the tags predicted by different orders are not identical, choose more probable one. Therefore, all tested algorithms we train in both forwards (F) and backwards (B) orders. Achieved values of evaluation metrics are shown in Table II.

Table II

Evaluation of taggers

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tagger** | **Accuracy** | | **Precision** | | **Recall** | |
| **F** | **B** | **F** | **B** | **F** | **B** |
| Unigram | 0.86 | 0.86 | 0.685 | 0.676 | 0.619 | 0.619 |
| Bigram | 0.38 | 0.39 | 0.754 | 0.726 | 0.30 | 0.289 |
| Trigram | 0.20 | 0.19 | 0.771 | 0.724 | 0.166 | 0.139 |
| Combined | 0.89 | 0.88 | 0.735 | 0.721 | 0.692 | 0.682 |
| HMM | 0.71 | 0.73 | 0.758 | 0.744 | 0.594 | 0.575 |
| Perceptron | 0.95 | 0.95 | 0.824 | 0.822 | 0.794 | 0.791 |

As it can be seen from Table II, Unigram, Bigram and Trigram taggers did not give such good results considering that they work separately, more precisely, backoff taggers were not selected during their training. However, for this reason, a tagger that combines these three algorithms was created. Some improvement in accuracy has been shown when these three algorithms work together.

Perceptron tagger gave the best results compared to all others. This was expected considering that the algorithm itself works according to the principle of a simple neural network.

1. Conclusion

Within this research, models were improved due to preprocessing of original corpus and applying grammar rules, although metric results were not changed. New and improved models that consider following words in a sentence are created and their metric result proved to be pretty good (they do not differ much from the models trained on the original corpus). Existing implementation from NLTK library uses algorithms that were not that much demanding to train and test, but they have shown good values for evaluation metrics.

Further work could go into three directions:

* 1. It could include analysis of examples in which forwards method is better and in which backwards method is better. In those situations where they output different tags, the one, which in that specific case gives better results, should be used.
  2. It could include possible implementation of hierarchical tagger. The similar idea as with backwards and forwards tagger could be used. Tags could be divided into three levels - type, subtype and case and each one of this level could be determined by a separate model – the one that gives the best results for that word and that level.
  3. It could include development of transformer language model (deep learning model) for tagging words in sentences written in the Serbian language.

References

1. SrpLemKor, <http://www.korpus.matf.bg.ac.rs/SrpLemKor/>, (last access on 3.5.2023)
2. Corpora of Contemporary Serbian Language, <http://korpus.matf.bg.ac.rs/prezentacija/uputstvo.html#aurora>, (last accessed on 3.5.2023.)
3. srWaC – Serbian web corpus, <http://nlp.ffzg.hr/resources/corpora/srwac/>, (last access on 3.5.2023)
4. N. Ljubešić, F. Klubička, “{bs,hr,sr}WaC - Web Corpora of Bosnian, Croatian and Serbian”, Proceedings of the 9th Web as Corpus Workshop (WaC-9), pp. 29–35, Gothenburg, Sweden, 2014.
5. MULTEXT-East Croatian part-of-speech tagset (version 5), <https://www.sketchengine.eu/multext-east-croatian-part-of-speech-tagset/>, (last access on 3.5.2023)
6. B. Milovanovic, R. Stanković, “PoS Tagging for Serbian language using NLTK”, Proceedings of 7th International Conference on Electrical, Electronic and Computing Engineering IcETRAN, pp. 3-7, 2020.
7. N. Ljubešić, F. Klubička, Ž. Agić and I. P. Jazbec, “New Inflectional Lexicons and Training Corpora for Improved Morphosyntactic Annotation of Croatian and Serbian”, Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), pp. 4264–4270, Portorož, Slovenia, 2016
8. GitHub - Serbian-AI-Society/SrbAI: Python library for Serbian Natural language processing (NLP), <https://github.com/Serbian-AI-Society/SrbAI>, (last accessed 3.5.2023.)
9. N. Ljubešić, D. Lauc, “BERTić -- The Transformer Language Model for Bosnian, Croatian, Montenegrin and Serbian”, Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing, pp. 37-42, Kyiv, Ukraine, 2021
10. N. Ljubešić and K. Dobrovoljc, “What does Neural Bring? Analysing Improvements in Morphosyntactic Annotation and Lemmatisation of Slovenian, Croatian and Serbian”, 7th Workshop on Balto-Slavic Natural Language Processing, pp.29–34, Florence, Italy., 2019
11. T. Djordjevic, S, Stojkovic, “A Tool for Sentence Syntax Structure Markup for The Serbian Language”, Proc. 8th International Conference on Electrical, Electronic and Computing Engineering (IcETRAN 2021), pp. 485-499, Ethno Village Stanisici, Republic of Srpska, 2021
12. D. Jurafsky and J. H. Martin, “Speech and Language Processing”. Copyright © 2023. All rights reserved. Draft of January 7, 2023.
13. M. Kosanović, I. Đurović, D. Dimitrijević and S. Stošović, “Implementacija algoritma za etiketiranje reči u srpskom jeziku u programskom jeziku Python”, INFOTEH-JAHORINA Vol. 15, pp.694-699, 2016
14. S. Tubić. “SL PERCEPTRON ALGORITHM”, Elektrotehnički fakultet, Univerzitet u Beogradu, 2014.