Article

Parallel Bidirectionally Pretrained Taggers as Feature Generators

Ranka Stanković [1,*,†] **, Mihailo Škorić** [1,†] **and Branislava Šandrih Todorović** [2]

1 Faculty of Mining and Geology, University of Belgrade, Djusina 7, 11120 Belgrade, Serbia; mihailo.skoric@rgf.bg.ac.rs
2 Faculty of Philology, University of Belgrade, Studentski Trg 3, 11000 Belgrade, Serbia; branislava.sandrih@fil.bg.ac.rs
* Correspondence: ranka.stankovic@rgf.bg.ac.rs
† These authors contributed equally to this work.

Abstract: In a setting where multiple automatic annotation approaches coexist and advance separately but none completely solve a specific problem, the key might be in their combination and integration. This paper outlines a scalable architecture for Part-of-Speech tagging using multiple standalone annotation systems as feature generators for a stacked classifier. It also explores automatic resource expansion via dataset augmentation and bidirectional training in order to increase the number of taggers and to maximize the impact of the composite system, which is especially viable for low-resource languages. We demonstrate the approach on a preannotated dataset for Serbian using nested cross-validation to test and compare standalone and composite taggers. Based on the results, we conclude that given a limited training dataset, there is a payoff from cutting a percentage of the initial training set and using it to fine-tune a machine-learning-based stacked classifier, especially if it is trained bidirectionally. Moreover, we found a measurable impact on the usage of multiple tagsets to scale-up the architecture further through transfer learning methods.

Keywords: annotation; natural language processing; feature extraction; composite structures; part of speech

1. Introduction

Automatic assignment of elements of the Part-of-Speech (POS) category, such as nouns, verbs, adjectives, etc., to words or text chunks is well-known and is one of the most common Natural Language Processing (NLP) tasks. POS-tagging has applications in many NLP pipelines, including Document Classification, Named Entity Recognition, Sentiment Analysis, and Question answering [1]. Task-performing software, usually labeled POS-tagger, can be either rule-based using lookup-tables, dictionaries, and/or extracted linguistic rules [2]; stochastic-based using various machine learning technologies from support vector machines [3] to the recurrent neural network (RNN) [4] and Deep Neural Networks [5]; or hybrid, combining the two, with an example being the TreeTagger, a software employing both lookup tables and dictionaries with the Hidden Markov Models (HMM) stochastic approach [6].

For Serbian, TreeTagger has long been deemed the optimal tagging approach [7]. It relies on rich lexical data being available for language [8], but the spaCy framework [9], which uses a more contemporary machine learning technology seems to be a promising alternative [10]. Experiments with other taggers, such as the Natural Language Toolkit (NLTK) library POS-tagger [11] have not produced satisfying results [12]. Stanford's new conditional random fields (CRF)-based Stanza system [13] also utilizes additional pre-trained resources, like word embedding models and dependency trees, in order to achieve an improved performance. It was remastered for Serbian under the name Classla [14] and achieved quite good, but not out of the ballpark, results for POS-tagging.

1.1. Related Work

The first comparative analysis of POS-taggers applied to texts in Serbian was conducted using 10-fold cross validation and compared three different taggers [15]. The study reported a high performance by both TnT (Trigrams'N'Tags) [16] and the TreeTagger [6] tool (accuracy on the training set, 93.86% and 91.78%, respectively), whilst SVMTool [3] was shown to have a somewhat better performance in certain special cases. The author also noted the possibility of combining different tagging methods and tools and integrating these with other NLP environments to open a wide area for further investigations and experiments regarding these solutions.

Another study [7] experimented with three different taggers, namely the TreeTagger, TnT, and Unitex [17], for the purpose of annotating the Corpus of Contemporary Serbian (SrpKor2013) [18]. After an exhaustive evaluation and discussion, the author named TreeTagger as the best candidate and applied it for annotation of the SrpKor2013. This choice was justified by the fact that TreeTagger assigns the information about the lemma, which is an especially desired feature in the task of corpus tagging.

We also recently compared several versions of TreeTagger and spaCy for Serbian [19] and found that each system has specific pros and cons. While spaCy's RNN-based contemporary approach to training gives it a slight edge when there is a high token overlap between the training and test sets, it underperforms compared with TreeTagger when dealing with unfamiliar words. When TreeTagger is in such a situation, a dictionary look-up is performed, but spaCy lacks such an instrument in its out-of-the-box form. The main question remaining is what is the best way to combine the pros of these taggers to get the optimal results.

An experiment combining multiple POS-taggers for Swedish [20] concluded that most types of combination algorithms work better than the best standalone tagger. Another combination attempt was done for Icelandic [21], where the accuracy was improved by nearly two percent using multiple taggers and simple combination algorithms. These papers established a basic set of combining techniques. The baseline is *voting*, where the tag with the most votes from different taggers is picked. The problem with this approach is that it does not work at all when only two taggers are available, and even if there are more taggers available, there is a high chance of a tie occurring, which cannot be resolved without the implementation of another algorithm. A slight improvement is offered through *weighted voting*, which applies weights to different taggers or tags, so the chance of a tie is lowered. Other techniques, such as *bidding* and *scoring*, require taggers to output tag probabilities. In *bidding*, the final tag is the one with the highest probability among all tags from all taggers, while in *scoring*, probabilities for tags are summed up among taggers, and the tag with the highest sum is selected.

The latest milestone in the development of composite POS-taggers is the *stacking* technique. This approach is used to eliminate the main cause of the low performance of composite taggers—the low-performing standalone tagger, offering high probabilities for its choice. This approach adds another supervised classifier on top of the results of standalone taggers, and if the classifier is well trained, it can learn that the low-performing tagger is not to be trusted. The initial experiment using probabilities as features [22] explored the composite system, which uses probabilistic outputs (for each tag and for each standalone tagger) of several standalone taggers as features for an HMM stacked classifier, which produces the final tag output.

1.2. Motivation

Based on these previously conducted experiments, we concluded that, in general, the composite tagging approach has two fundamental requirements:

1. Multiple training and tagging technologies must be available.
2. These existing technologies must differ in their approaches or at least yield different results (ideally including unique correct annotations).

The first requirement is undoubtedly true, as different POS-taggers achieve good results, but for the second requirement requires the testing of different taggers and the attempt to procure which ones will contribute the most in a composite environment. Our preliminary analysis for this experiment, which was in the form of a comparison of mislabeled tokens between TreeTagger and spaCy, supported the hypothesis that our existing taggers do provide different results, despite their similar levels of performance. TreeTagger correctly guessed about 8% of tags that spaCy missed, while SpaCy correctly guessed about 3% of tags that TreeTagger did not, which is an indicator that further improvement is possible.

We performed another experiment to uphold this point: since TreeTagger has a built-in system for outputting probabilities for each tag (the side effect of HMM usage), we devised a simple composite system that produces a single tag from these two taggers (Figure 1). If the taggers disagree on an assigned tag, the final decision is made based on TreeTagger's probability output for the selected tag. A probability of over 50% gives it a higher weight, and that tag is chosen as the final answer; otherwise, spaCy's selected tag is picked. The application of this algorithm found another 0.55% accuracy improvement over the initial TreeTagger and another 1.03% improvement over the spaCy result, which was a green flag for further inspection of this methodology.
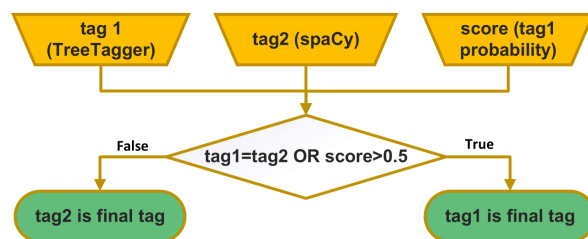

Figure 1. Simple composition algorithm based on the outputs of the TreeTagger and spaCy POS-taggers, together with TreeTagger's probability outputs.

1.3. Research Questions, Aims, Means, and Novelty

In this paper, we propose a novel stacked language-independent classifier architecture. We use probabilistic outputs of multiple bidirectionally pretrained POS-taggers (spaCy, TreeTagger and Stanza) as inputs to an improved neural network classifier. The approach is then evaluated on texts in Serbian with the aim of answering several research questions.

RQ1　Is bidirectional training of POS-taggers an appropriate way to create new, independent models of unique value?

RQ2　Can composition-based POS-taggers outperform standalone pretrained taggers?

RQ3　Is the stacked classifier the optimal way to combine outputs of pretrained POS-taggers in order to improve the overall performance and stability?

RQ4　Is bidirectional training of POS-taggers through dataset augmentation an easy and appropriate way to further improve the results of the stacking architecture as compared with standalone taggers?

To that end, we used the following steps:

1.　Procure a balanced dataset of texts, preannotated for POS using at least one tagset;
2.　Test several POS-taggers that are able to output the probabilities necessary for our stacked architecture, train them bidirectionally using dataset augmentation, and determine which ones would contribute most in a composite environment by measuring their mutual entropy and their ability to produce uniquely correct annotations;
3.　Train composition-based classifiers that use probabilistic outputs of selected pretrained standalone taggers as features;
4.　Evaluate the relative performance of standalone taggers, unidirectionally and bidirectionally built composite taggers, as well as several other combination techniques.

The main contributions of this paper are the following:

1.　A novel bidirectional training architecture, where the input is not only the verticalized training set but its augmented, inverted variant as well;
2.　A novel stacked tagger architecture combining off-the-shelf taggers;
3.　The first bidirectional combined POS-tagger for the Serbian language, which outperforms the state-of-the-art taggers [19];
4.　An up-to-date comparative analysis of the state-of-the-art taggers.

This article is structured as follows. The second section introduces the dataset that was used for the training and evaluation of POS-taggers for this research and explores the process of initial tagger selection. It also tackles the methodology used for the bidirectional training of taggers and explores training, tagging, and evaluation pipelines. The third section presents the results obtained through the evaluation process, followed by a discussion of the results and concluding remarks, together with plans for future research.

2. Materials and Methods

Training and testing of all presented taggers were done using a publicly available annotated corpora for Serbian, SrpKor4Tagging (https://live.european-language-grid.eu/catalogue/corpus/9295, accessed on 11 March 2022), with 342,804 annotated tokens. Roughly one-third of the dataset tokens originated from literary texts (novels and novel excerpts), while the rest originated from nonliterary texts (news articles, textbooks, and administrative texts) [19].

The dataset is segmented into sentences and pretagged with two different tagsets—the Universal POS tagset consisting of seventeen classes (https://universaldependencies.org/u/pos, accessed on 11 March 2022), which was mainly used in this experiment, and the SrpLemKor POS tagset consisting of sixteen classes (www.korpus.matf.bg.ac.rs/SrpLemKor/tagset.html, accessed on 11 March 2022), which was developed for Serbian morphological dictionaries [SMD] [23], in accordance with traditional, descriptive Serbian grammar. The Unitex corpus processing system and SMD were used to preannotate tokens, which were manually disambiguated. For the purposes of this experiment, the Universal POS tagset was used for the training and evaluation of all taggers, and sentences were randomly shuffled to avoid bias, during both the initial tests and the subsequent evaluation.

Among the 342,804 corpus words, there are 33,343 types (different tokens) and 14,588 different lemmatized forms. The frequency distribution within Universal POS tagset is as follows: NOUN (88832), PUNCT (48993), ADJ (35924), VERB (32154), ADP (30474), AUX (18591), CCONJ (16454), DET (15598), PART (11482), ADV (10881), SCONJ (10751), NUM (9729), PRON (6008), PROPN (4487), X (2393), and INTJ (52). The frequency distribution within the SrpLemKor tagset is as follows: N (93281), V (50745), A (35924), PUNCT (33275), PREP (30474), CONJ (27205), PRO (21606), SENT (15718), PAR (11458), ADV (10905), NUM (9539), ABB (2016), X (288), RN (190), PREF (127), and INT (52).

To maximize the impact of the composite architecture, we undertook a thorough inspection of POS-tagger candidates. We wanted the taggers in the composite system to be as good as possible (performing well as standalone taggers in the first place) and as unique as possible. This is because there is no point in combining taggers that yield similar results—their outputs should be diverse. On the other hand, there should also be as many of them as possible to achieve optimal results [22].

During the initial testing, we trained five POS-tagging systems (TreeTagger, spaCy, RNNTagger, Stanza, and NLTK) for the Universal POS tagset on 90% of our prepared dataset using default out-of-the-box settings and annotated the remaining 10% with each tagger for testing purposes. We supplemented TreeTagger with a publicly available version of the previously mentioned SMD, dubbed SrpMD4Tagging (https://live.european-language-grid.eu/catalogue/lcr/9294, accessed on 11 March 2022), as the lexicon [19,23,24], and we supplemented the Stanza tagger with a pretrained model of word embedding vectors available at the Hugging Face page (https://huggingface.co/stanfordnlp/stanza-sr/tree/main/models/pretrain, accessed on 11 March 2022). As probabilistic outputs were used for testing, all taggers were required to output a probability for each tag/class. For TreeTagger and NLTK (using the multinominal

Naive Bayes), probabilities were readily available, and for the three RNN-based taggers, RNNTagger, spaCy, and Stanza, a Softmax (normalized exponential) function was used.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{1}$$

where x_i is an input vector and x_j is and output vector. This was applied to the output tensors to procure the actual probabilities. Using this method, a vector of probabilities was output by all taggers for each token in the tagged test set. To test how unique each tagger's outputs were, we used these vectors to calculate the average Cross-Entropy Loss [CEL] using the formula

$$\text{CEL}(\hat{y}, y) = -\sum \hat{y} \log y \tag{2}$$

where $\hat{y}$ is the predicted vector and y is the target vector for every token for every pair of taggers. This was used to detect pairs giving results that were too similar to justify the inclusion of both taggers in the composite system. These cross-results are shown in the upper half of Table 1.

Table 1. Cross-Entropy Loss (CEL) between probabilistic outputs of tagger pairs (top), and the average CEL, test set CEL, uniqueness score, and number of unique correct annotations (UCA) for each tagger with outlying values presented in bold (bottom).

	TreeTagger	spaCy	RNNTagger	Stanza	NLTK	Test Set
TreeTagger		6.15	6.81	6.40	7.79	4.25
spaCy	6.15		6.09	3.35	10.14	5.19
RNNTagger	6.81	6.09		5.27	5.82	6.84
Stanza	6.40	3.35	5.27		9.93	5.71
NLTK	7.79	10.14	5.82	9.93		9.05
average CEL	6.79	6.43	6.00	6.24	8.42	
test set CEL	4.85	5.19	6.84	5.71	**9.05**	
uniqueness	1.94	1.24	**−0.84**	0.53	**−0.63**	
UCA	1164	538	**17**	391	**0**	

For each tagger, we also calculated the total average CEL, the CEL against correct annotations (test set CEL), and the difference between the two (uniqueness score) as well as counting unique correct annotations (UCA). Each time a tagger correctly predicted a POS tag for a token that no other tagger predicted correctly, it was awarded one UCA point. These additional metrics are shown in Table 1 (bottom half) and indicate how good and how unique each of these taggers is and which of them satisfy the composite system requirements. Suitable taggers had to have a high average CEL and a low test set CEL, yielding a high uniqueness score and a number of UCA points, the higher the better.

NLTK was immediately identified as an outlier with a negative uniqueness score (−0.63), despite a very high average CEL (8.42), and most importantly, it gained no UCA points. RNNTagger, Stanza, and spaCy were all built upon the same RNN technology and trained on the same dataset, so their results were bound to be similar, as demonstrated by the smallest CEL between these taggers shown in Table 1. However, the RNNTagger was the only tagger other than NLTK with a negative uniqueness score (−0.84), with only seventeen UCA in the set of over 30,000 tokens. Further inspection showed that most of its correct annotations were also correctly annotated using the spaCy or Stanza taggers but not vice-versa.

Upon reflecting on the results, we decided to drop NLTK and RNNTagger from further experiments and focus on bidirectional training of the TreeTagger, spaCy, and Stanza taggers and their integration into the composite environment.

2.1. Bidirectional Training

Bidirectional training was popularized primarily via the phenomenon of Bidirectional Encoder Representations from Transformers (BERT) [25] and, later, its derivatives. While

BERT uses bidirectional training for maximizing context awareness (using both left and right contexts in the creation of context-aware word embeddings), we used it to train separate, standalone models of our POS-taggers, which we later combined in a composite bidirectional architecture.

To test if this is feasible, we trained three new taggers on the inverted dataset (generated simply through the inversion of our preannotated token array) with no other changes to the training procedures. The only difference in these inverted taggers (−I) was that they were supplied with inverted texts during tagging, and the obtained results needed to be inverted again to preserve the original token order. To tag the sentence *Hello world!* using an inverted tagger, you need to supply it with the following list of tokens:

1. !
2. world
3. Hello

and simply invert the resulting array of tags or tag probabilities. As one of the goals was to test our hypothesis that these inverted taggers satisfy the composite environment requirements and perform well as standalone taggers, we recreated the tests from the previous section, and the results are presented in Table 2. The only addition is the bottom row, which shows the taggers' bidirectionally unique correct annotations (BUCA). Each time a tagger correctly annotates a token missed by its inverted counterpart (and vice-versa) it gains a BUCA point, and a higher BUCA score represents more effective bidirectional training of the tagger.

Table 2. CEL between tagger pairs, including the inverted taggers (top), average CEL, test set CEL, uniqueness score, and UCA and BUCA for each tagger (bottom) with some outlying values presented in bold.

	TreeTagger	spaCy	Stanza	TreeTagger-I	spaCy-I	Stanza-I
TreeTagger		6.15	6.40	**1.54**	6.18	6.27
spaCy	6.15		3.35	6.41	4.12	4.00
Stanza	6.40	3.35		6.53	4.35	3.19
TreeTagger-I	**1.54**	6.41	6.53		6.43	6.18
spaCy-I	6.18	4.12	4.35	6.43		5.06
Stanza-I	6.27	4.00	3.19	6.18	5.06	
Average CEL	5.31	4.81	4.76	5.42	5.23	4.94
test set CEL	4.25	5.19	5.71	4.58	5.17	**6.05**
uniqueness	1.06	−0.38	−0.95	0.84	0.06	−1.31
UCA	57	59	57	42	59	48
BUCA	444	1038	1002	394	1064	695

We found that inverted taggers not only have actual unique contributions but, in some cases, outperform the standard taggers (spaCy-I against spaCy BUCA). Even in the case of TreeTagger, which had a very modest CEL of 1.54 (TreeTagger row, TreeTagger-I column of Table 2), UCAs were still found. Taggers also showed only small differences in the average CEL and test set CEL with no specific outliers. The highest recorded CEL against the test set was 6.05 from the Stanza-I tagger, which was still better than the results from NLTK and RNN-Tagger presented in the previous section (6.84 and 9.05, respectively). Finally, by comparing the UCAs between all six taggers (UCA row in Table 2), we found a tendency toward an equal dispersion, making each of these six taggers equally suitable for the composite environment.

2.2. Training and Tagging Pipelines

For the purposes of this research, we devised one algorithm for bidirectional training of the proposed stacked architecture and another for tagging documents using the trained resources. The training pipeline is composed of the bidirectional training of the standalone

taggers and the fine-tuning of a stacked classifier using their respective probabilistic outputs on a token-level. This algorithm is depicted in Figure 2 and can be divided into four steps.
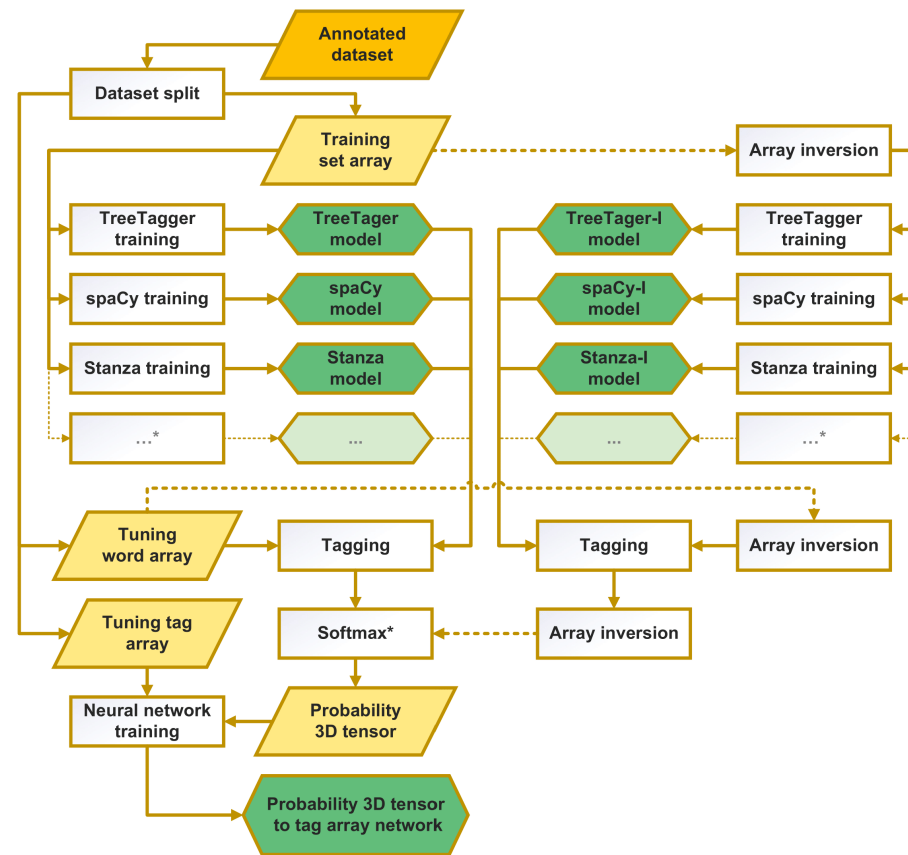
Figure 2. Training algorithm for the composite model with the right side (connected via dashed line) being the bidirectional expansion. The architecture is vertically scalable (uncontainable) with the addition of any number of new taggers bidirectionally—represented with . . . *.

(1) Training and tuning set formation. First, we group annotated sentences into two chunks (we shuffled them beforehand and used a four to one ratio, although this is not obligatory). We used the bigger chunk as the training set and the smaller chunk as a tuning set, splitting the data vertically into two separate arrays: one containing only tokens and the other containing correct POS tags for those tokens. The first step resulted in three arrays: the training set containing both tokens and POS tags; one tuning set containing only tokens; and the other tuning set containing only the POS tags.

(2) Standalone tagger training. We used the previously built training set to train the selected standalone taggers. The architecture is vertically scalable, as presented in Figure 2, where . . . * denotes any number of additional taggers, with the only requirement being that they produce probabilistic outputs. We used the TreeTagger, spaCy, and Stanza POS-taggers for the experiment and trained them bidirectionally with standard out-of-the-box training for each tagger. Taggers do not need to use the same tagset, as every tag for every tagger becomes a new, separate feature (note, however, that this will render their standalone evaluation for resulting tagset pointless). Basically, the second step results in standalone tagger models trained on the bigger portion of the annotated dataset.

(3) Creation of the probability tensor. This tensor was formed using the tuning token array created in step 1 and the pretrained standalone taggers created in step 2. It should be noted that it is also possible to use other pretrained taggers, but they should be trained on the set not overlapping the tuning set to avoid bias. We took probabilities for each tag from each tagger used (inverted taggers tagging the inverted array of tokens) and for each token in the tuning array (by applying the SoftMax function to normalize outputs of RNN-based

taggers). These probability matrices (for each token and for each tag) for each standalone tagger were concatenated into a single tensor object, the output of this step.

(4) Stacked classifier training. Stacked classifier training was performed using the tuning tag array created in the first step and the probability tensor created in the previous step. Probabilities from the tensor beaome features of each token when training the classifier, while correct tags for those tokens (tuning tag array) became classes to be predicted. For the stacked classifier, we used a single perceptron, the simple artificial neural network with one input layer (one node for every feature) and fully connected with a single output layer (one node for every tag). The final product of this step was a neural network that could transform combined probabilistic outputs of all standalone taggers into a single tag for each token.

We created the tagging architecture (Figure 3) in accordance with the previously detailed training architecture. It requires the trained stacked classifier and all the pretrained taggers used in its preparation and consists of three steps.

(1) Preprocessing. In this step, we prepared an array of tokens to be tagged.

(2) Feature generation. We bidirectionally tagged the array prepared in the previous step. As is usual for inverted taggers, the inverted array was tagged, and the output was inverted once again. The Softmax function was applied to transform the outputs of the RNN-based taggers into probabilities.

(3) Using the stacked classifier. Based on the probabilities of each tag from each tagger from the previous step, a single tag was output for each token using the trained perceptron-based stacked classifier.
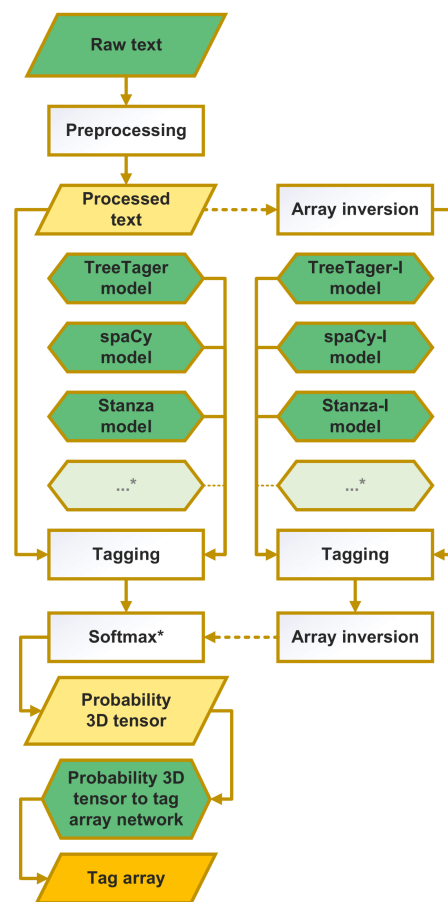

Figure 3. Tagging algorithm used in our composite model with the right side (connected via dashed line) being the bidirectional expansion. The architecture is vertically scalable (uncontainable) with an addition of any number of new taggers bidirectionally—represented with . . . *.

2.3. Evaluation

For the evaluation, we used the aforementioned dataset and the training and tagging architecture to procure average weighted f_1-scores for tagging with the Universal POS tagset. Since we had a stacked classifier, we used a five-fold over five-fold nested cross-validation for the evaluation with 25 evaluation runs conducted in total (Figure 4). Since a 4 to 1 ratio was used for splitting into both training and testing as well as training and tuning sets, in each run, 64% of the dataset was used to train the standalone taggers, 16% was used to train the stacked classifier, and 20% was used for testing the performance of all taggers.
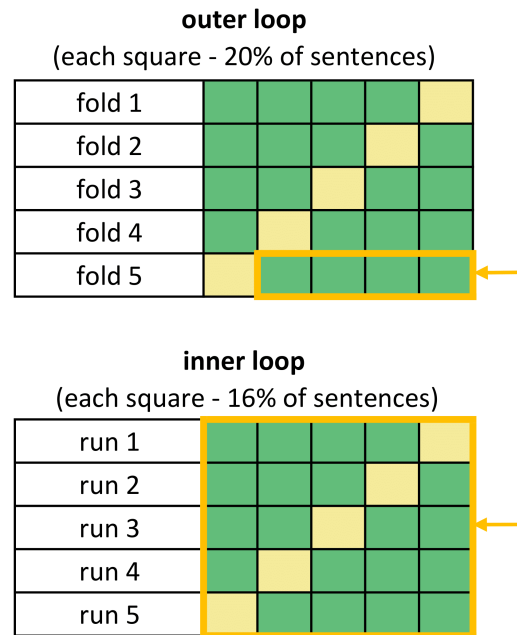

Figure 4. Outer and inner loops of nested cross-validation. In the outer loop, 20% of the annotated sentences were used as the test set. In the inner loop, the remaining 80% were split in a 4:1 ratio, resulting in 64% for the training set and 16% for the tuning set for each run.

For each of these runs, we trained six different standalone taggers (three standard and three inverted) and two stacked classifiers (one using all six taggers and one using only the three standard ones). For each run, we also tested three additional rule-based composition algorithms: *voting*, *bidding* and *scoring*.

3. Results

The evaluation results (in the form of weighted f_1-scores) for each standalone tagger were averaged out for each outer cross-validation fold and are presented in Table 3. The upper section of the table shows the results grouped by tagger and fold, while the lower section of the table shows the average, minimum and maximum f_1-scores, as well as the performance stability score (calculated as the ratio of the minimum and maximum scores) for each tagger.

Table 3. Comparison of the averaged weighted f_1-scores for each of the outer five folds and each of the six tested standalone taggers (top), and a comparison of their average, minimal, and maximal scores as well as their stability, calculated as the ratio of minimum to maximum scores (bottom).

Fold	spaCy	spaCy-I	TreeTagger	TreeTagger-I	Stanza	Stanza-I
#1	0.9465	0.9483	0.9519	0.9539	0.9389	0.9275
#2	0.9457	0.9467	0.9538	0.9522	0.8760	0.8911
#3	0.9470	0.9428	0.9550	0.9532	0.8819	0.9254
#4	0.9436	0.9467	0.9556	0.9540	0.9386	0.8839
#5	0.9448	0.9458	0.9547	0.9535	0.8741	0.8431
average	0.9455	0.9461	0.9542	0.9534	0.9019	0.8942
minimum	0.9436	0.9428	0.9518	0.9522	0.8741	0.8431
maximum	0.9470	0.9483	0.9556	0.9540	0.9389	0.9275
stability	0.9964	0.9942	0.9961	0.9981	0.9310	0.9090

For each of the five folds, we also calculated a total average score across all standalone taggers and extracted the best score (among the standalone taggers) in each fold to be used as a baseline for testing the relative performance levels of the four composite methods (*voting*, *bidding*, *scoring* and *stacking*). The results of these tests are presented in Table 4 as a comparison of the baseline and unidirectionally built composite methods and in Table 5 as a comparison of the baseline and bidirectionally built composite methods. The lower sections of these two tables also present the same compiled metrics mentioned in the description provided for Table 3.

Table 4. Comparison of the derived baseline (left) and averaged weighted f_1-scores for each of the outer five folds and each of the four tested composition methods built on the unidirectionally trained standalone taggers (right). The best result for each fold and each additional metric is shown in bold.

Fold	Standalone Average	Standalone Best (Baseline)	Voting	Bidding	Scoring	Stacking
#1	0.9458	0.9519	0.9584	0.9608	**0.9718**	**0.9718**
#2	0.9252	0.9538	0.9495	0.9589	0.9677	**0.9700**
#3	0.9280	0.9550	0.9490	0.9597	0.9668	**0.9715**
#4	0.9459	0.9556	0.9503	0.9597	0.9715	**0.9721**
#5	0.9245	0.9547	0.9612	0.9611	0.9677	**0.9715**
average	0.9339	0.9546	0.9537	0.9600	0.9691	**0.9714**
minimum	0.9245	0.9538	0.9490	0.9589	0.9668	**0.9700**
maximum	0.9459	0.9556	0.9612	0.9611	0.9718	**0.9721**
stability	0.9734	0.9964	0.9881	0.9955	0.9979	**0.9989**

Table 5. Comparison of the derived baseline (left) and averaged weighted f_1-scores for each of the outer five folds and each of the four tested composition methods built on the bidirectionally trained standalone taggers (right). The best result for each fold and each additional metric is shown in bold.

Fold	Standalone Average	Standalone Best (Baseline)	Voting	Bidding	Scoring	Stacking
#1	0.9445	0.9539	0.9614	0.9724	**0.9756**	**0.9756**
#2	0.9276	0.9538	0.9536	0.9697	0.9742	**0.9752**
#3	0.9342	0.9550	0.9559	0.9721	**0.9755**	0.9754
#4	0.9371	0.9556	0.9599	0.9721	0.9745	**0.9751**
#5	0.9193	0.9547	0.9651	0.9741	0.9736	**0.9762**
average	0.9325	0.9546	0.9592	0.9721	0.9747	**0.9755**
minimum	0.9193	0.9538	0.9536	0.9697	0.9736	**0.9751**
maximum	0.9445	0.9556	0.9651	0.9741	0.9756	**0.9762**
stability	0.9774	0.9961	0.9873	0.9977	0.9949	**0.9978**

The results presented in Tables 3–5 are compiled and visualized as a heatmap in Figure 5. The upper section of the figure depicts a comparison of all standalone taggers and unidirectionally built composite methods, while the lower section does the same but for bidirectionally built composite ones. The relative improvements visualized in the heatmap were also quantified as percentages of the error reduction using the formula

$$\text{Error Reduction} = \frac{a_1 - a_0}{1 - a_0} \tag{3}$$

where a_0 is the initial accuracy and a_1 is the supposedly improved one. The obtained results are numerically presented as in Table 6, where the left side shows the error reduction achieved using the unidirectionally-built composite methods, and the right side shows the same for the bidirectionally-built methods.
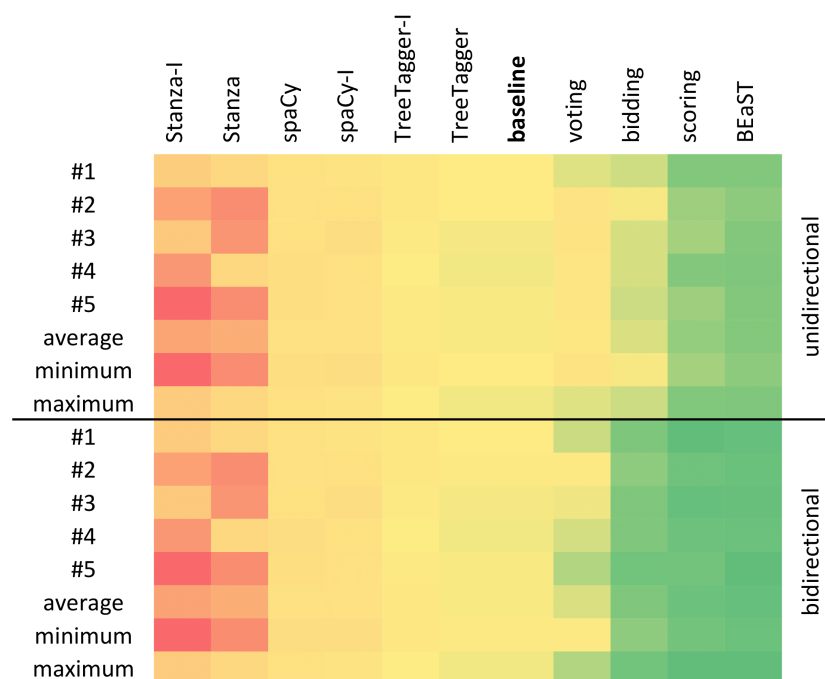

Figure 5. Heatmap depicting differences in achieved f_1-scores for all standalone taggers and unidirectionally built composite ones (upper) and bidirectionally built ones (lower). Yellow indicates the average f_1-scores, red (left-side) indicates the relative decrease, and green (right-side) indicates the relative increase in the score.

Table 6. Relative error reduction achieved using the unidirectionally based composite methods (left) and bidirectionally built methods (right) for each of the five outer folds (upper) and the compiled metrics for each method (lower). Greatest improvements over the baseline are shown in bold.

Fold	Voting	Bidding	Scoring	Stacked	Voting	Bidding	Scoring	Stacked
#1	13.51%	18.50%	41.37%	41.37%	16.27%	40.13%	**47.07%**	**47.07%**
#2	−9.31%	11.04%	30.09%	35.06%	−0.43%	34.42%	44.16%	**46.32%**
#3	−13.33%	10.44%	26.22%	36.67%	2.00%	38.00%	**45.56%**	45.33%
#4	−11.94%	9.23%	35.81%	37.16%	9.68%	37.16%	42.57%	**43.92%**
#5	14.35%	14.13%	28.70%	37.09%	22.96%	42.83%	41.72%	**47.46%**
average	−1.34%	12.67%	32.44%	37.47%	10.87%	39.04%	44.72%	**46.51%**
minimum	−13.33%	9.23%	26.22%	35.06%	2.93%	36.61%	44.77%	**47.91%**
maximum	14.35%	18.50%	41.37%	41.37%	21.40%	41.67%	45.05%	**46.40%**

The comparison of the results achieved using unidirectionally-built composite methods (Table 4) and bidirectionally-built composite methods (Table 5) was recompiled to

indicate their mutual differences and the results are displayed in Table 7. The table shows relative f_1 score improvements achieved using the bidirectional architecture over uthe nidirectional one (upper section) and the relative average, minimum, and maximum values and the stability improvement (lower section).

Since the ambiguous words (ones that can be annotated with several different tags) are a common issue in POS-tagging, we analyzed a few well known difficult cases for Serbian to procure how different taggers resolved them. Results for three characteristic ambiguous hard cases we tested are presented and discussed in the Appendix A.

Table 7. Relative improvements achieved using the bidirectional architecture over the unidirectional one (upper section) and the relative average, minimum, and maximum values and the stability improvement (lower section).

Fold	Voting	Bidding	Scoring	Stacking
#1	7.21%	29.59%	15.60%	13.48%
#2	8.12%	26.28%	20.12%	17.33%
#3	13.53%	30.77%	26.20%	13.68%
#4	19.32%	30.77%	10.53%	10.75%
#5	10.05%	33.42%	18.27%	16.49%
avg	11.65%	30.17%	18.14%	14.35%
min	7.21%	26.28%	10.53%	10.75%
max	19.32%	33.42%	26.20%	17.33%
stability	0.08%	−0.22%	0.31%	0.10%

4. Discussion

From the results presented in Table 3 it can be seen that TreeTagger and spaCy gave similar scores, with TreeTagger having a slight edge in both the average performance (0.9542 and 0.9534 compared with 0.9455 and 0.9461) and stability scores (0.9961 and 0.9981 compared with 0.9964 and 0.9942), which aligns with our initial testing on this dataset. TreeTagger was trained using an additional lexical input in the form of annotated dictionaries with over two million word forms with their possible Part-of-Speech and lemma elements. Having dictionaries in the background, as expected, TreeTagger performed better than the others for unambiguous words that are in the dictionary, but spaCy was better at taking the context into account and resolving ambiguity. Another parallel was seen in the comparison of inversely-trained and standard taggers, where TreeTagger outperformed TreeTagger-I, and spaCy-I outperformed spaCy, both by a small margin of less than 0.1%. Stanza-based taggers matched this performance on several occasions (folds 1 and 4 for Stanza and folds 1 and 3 for Stanza-I), but they also displayed much greater instability with scores of 0.931 and 0.909, which subsequently resulted in less impressive averages of weighted f_1 scores (around 0.9). Nevertheless, inversely-trained taggers performed similarly to standardly-trained ones, as shown in our initial tests for this experiment, which affirms that inverted taggers are independent systems with unique value, giving the answer to RQ1:

RQ1: *Is bidirectional training of POS-taggers an appropriate way to create new independent models of unique value?*

As for the composition-based taggers, their clear advantage over the standalone ones is visible in both Tables 4 and 5, as well as in Figure 5. This is in line with reports from other studies, where most of the combination methods outperformed all of the standalone taggers. The only exception in our experiment was *voting* against TreeTagger and TreeTagger-I on folds 2, 3, and 4. The best overall f_1 score achieved using a standalone tagger was 0.9556 (TreeTagger, fold 4), while the best result using composite methods was 0.9762 (bidirectional stacked classifier, fold 5), and the worst score achieved using the best performing composition-based method was 0.97. As is visible in Table 6, three of the four

composition methods tested outperformed the baseline in all cases and in every category with an error reduction of up to 47.91% and 9.23% achieved for the worst case, giving a clear answer to RQ2:

RQ2: *Can composition-based POS-taggers outperform the standalone pretrained taggers?*

Our single-perceptron-based stacked classifier outperformed all other composition methods when it came to unidirectionally-built compositions, as is visible in Table 4. It was only matched by the *scoring* method on fold 1, but it displayed the highest average scores, the highest minimum and maximum scores, as well as the highest stability, reducing the error rate of the baseline to an average of 37.47%, while the next best method (*scoring*) provided an average error reduction of 32.44%.

When it came to comparing bidirectionally-built composite methods (Table 5), *stacking* was again the best performing method with an average error reduction of 46.51%, but the other methods improved slightly compared with the unidirectional architecture with *stacking* providing an average error reduction of 44.72%. While this could be a sign of *stacking* losing its improvement momentum with an increase in the number of taggers, it is more probable that it has a less steep slope and that it just performs better than *scoring* when dealing with less information. This is supported by its high stability, with a score of 0.9978 across five folds. Nevertheless, considering its other advantages, such as the possibility of using an open tagset and diminishing inputs from poor performance taggers, together with the fact that it outperformed other composition-based tagging methods in both of our tests in terms of both performance and stability, *stacking* was shown to be the best overall method (followed by *scoring*, and then *bidding*), thus answering RQ3:

RQ3: *Is the stacked classifier the optimal way to combine outputs of pretrained POS-taggers in order to improve the overall performance and stability?*

Relative improvements achieved through the use of the bidirectional architecture shown in Table 7 displayed substantial improvements over the unidirectional one. The bidirectional approach yielded improvements for all four tested methods in 100% of cases with average improvements ranging from 11.65% to 30.17%. It also resulted in increased minimum and maximum scores for all four methods and a stability improvement (up to 0.31%) for three out of the four tested methods. All of this undoubtedly answers RQ4:

RQ4: *Is bidirectional training of POS-taggers through dataset augmentation an easy and appropriate way to further improve the results of the stacking architecture over standalone taggers?*

After observing an increase in performance with an increased number of standalone taggers, we decided to try out one final run. As our dataset was annotated with two different tagsets, we took advantage of the possibility offered by the stacking environment to use them both. We trained another set of six standalone taggers on the second available tagset (SrpLemKor) and added the outputs of those taggers as features for the stacked classifier (increasing the number of features from 102 to 198). This final run resulted in an f_1 score of 0.9783, which is a higher score than the maximum score of any previous runs (0.9762) and a clear indication that further improvements are possible.

Considering this, future research will focus on a more detailed evaluation of the impact of training standalone models on additional tagsets (or other lexical information) and adding those models to the stacked architecture in pursuit of additional performance improvements.

Something that could also benefit future research is the extension of the evaluation to other languages, where bidirectional training could have either a lesser or a higher impact. More detailed research of the impact of the dataset size and ratios of its splitting into training, tuning, and testing sets, as well as the impact of adding standalone taggers to, or removing them from, the composition could also result in a better understanding of the

overall impact of the methodology for the task of POS-tagging as well as for other NLP tasks.

Since the results obtained through this particular method are completely reliant on the standalone taggers used for the training of the stacked classifier, they can be applied for any language in which these standalone taggers are trained, making the approach language-independent. Furthermore, the use of the Universal POS tagset, which has corpora available for at least 122 languages (https://universaldependencies.org/, accessed on 11 March 2022) makes the existing framework suitable for many other languages.

5. Conclusions

The paper presented a composite POS-tagging system based on bidirectional expansion as an alternative to the state-of-the-art stacked classifier. We tested the architecture on an annotated dataset for Serbian with three different annotation systems (TreeTagger, spaCy, and Stanza) and assessed the experiment as successful, confirming that

1. Bidirectional training of POS-taggers is an appropriate way to create new independent models with unique value;
2. Composition-based POS-taggers outperform standalone pretrained taggers in the majority of cases;
3. The stacked classifier is the optimal way to combine outputs of pretrained POS-taggers in order to improve the overall performance and stability;
4. Bidirectional training of POS-taggers through dataset augmentation is an easy and appropriate way to further improve the results of the stacking architecture over standalone taggers.

Our bidirectional, expandable, and stacked tagger (BEaST) yielded improvements in both performance and stability, providing an error reduction of up to 47.91%. Thus, we conclude that this is a good approach to training a POS-tagger on a limited preannotated dataset, especially considering the automatic expansion of the usable taggers domain through bidirectional training, which is readily available through automatic dataset augmentation. This is particularly viable for less-resourced languages, such as Serbian. As the system is scalable, new taggers can also always be added into the mix as long as a unique tuning dataset is used.

One final test also showed the possibility for further improvements by using more than one tagset for feature generation, which opens a gateway to new research on stacked classifier architecture expansion.

Author Contributions: Conceptualization, M.Š. and R.S.; methodology, M.Š. and R.S.; software, M.Š.; validation, R.S. and B.Š.T.; formal analysis, R.S.; investigation, B.Š.T.; resources, R.S.; data curation, R.S.; writing—original draft preparation, R.S. and M.Š.; writing—review and editing, all authors; visualization, M.Š.; supervision, R.S.; project administration, M.Š. All authors have read and agreed to the published version of the manuscript.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The annotated dataset used for the training and evaluation of models in this research, SrpKor4Tagging, is available at https://live.european-language-grid.eu/catalogue/corpus/9295 (accessed on 11 March 2022). The lexicon used for TreeTagger training, SrpMD4Tagging, is available at https://live.european-language-grid.eu/catalogue/lcr/9294 (accessed on 11 March 2022). The pretrained word vectors used to train Stanza taggers are available at https://huggingface.co/stanfordnlp/stanza-sr/tree/main/models/pretrain (accessed on 11 March 2022). The Python package that was used to reproduce this experiment, BEaSTagger, is publicly available as a Github repository at https://github.com/procesaur/BEaSTagger (accessed on 11 March 2022).

Abbreviations

The following abbreviations are used in this manuscript:

BEaST	Bidirectional, Extendable and Stacked Tagger
BERT	Bidirectional Encoder Representations from Transformers
BUCA	bidirectionally unique correct annotations
CEL	Cross-Entropy Loss
CRF	Conditional random filed
HMM	Hidden Markov Models
NLP	Natural language processing
NLTK	Natural Language toolkit
POS	Part-of-speech
RNN	Recurrent neural network
SMD	Serbian morphological dictionaries
UCA	unique correct annotations

Appendix A

For the analysis of (1) different taggers' precision with respect to a particular part of speech and (2) real-data examples (hard cases with ambiguous words), we randomly selected a set of sentences, making a text sample with 5119 tokens.

The precision of TreeTagger for this dataset was best for ADJ, DET, NOUN, NUM, PART, PROPN, SCONJ, and VERB, while TreeTagger-I was best for ADV and INTJ. SpaCy was best for ADP and CCONJ, while spaCy-I was best for PRON. SpaCy and TreeTagger-I had the same level of performance for AUX. For each tagger, we found cases where only one tagger had the correct tag assigned: spaCy (58), Stanza (27), and TreeTagger (130). More precisely, if we take direction into account, the following cases had only one tagger with the correct tag assigned: spaCy (12), spaCy-I (17), Stanza (4), Stanza-I (4), TreeTagger (5), and TreeTagger-I (11). This proves that the use of several taggers is justified.

Table A1 presents three ambiguous hard cases: *je*, *sam* and *tu* (totals are given in boldface).

Table A1. Three ambiguous hard cases: *je*, *sam* and *tu*.

Token		UPOS	spaCy	spaCy-I	Stanza	Stanza-I	TreeTagger	TreeTagger-I	high	BEaST
je		**126**	**120**	**120**	**120**	**120**	**120**	**120**	**119**	**119**
	AUX	120	114	119	120	115	120	120	114	119
	PRON	6	1	1	0	0	0	0	0	0
sam		**31**	**30**	**27**	**25**	**27**	**22**	**26**	**28**	**28**
	ADJ	1	0	0	0	0	0	0	0	0
	AUX	26	26	24	25	26	22	25	25	25
	DET	4	4	3	0	1	0	1	3	3
tu		**8**	**5**	**5**	**2**	**2**	**6**	**5**	**5**	**5**
	ADV	6	4	4	0	0	4	4	4	4
	DET	2	1	1	2	2	2	1	1	1
Grand Total		**165**	**155**	**152**	**147**	**149**	**148**	**151**	**152**	**152**

The corpus word *je* can be (1) the form of the auxiliary verb *jesam* (to be) (the clitic form of the third person singular, present tense) or (2) the form of the personal pronoun *ona* (she) (the clitic form of the genitive or accusative singular). The form *je* appeared 126 times: 120 times as the form of the auxiliary verb *jesam* (to be) and 6 times as a form of the pronoun

ona (she). All forms of the verb *jesam* were annotated correctly, but TreeTagger missed all pronouns and Spacy annotated only one correctly.

The form *sam* can be (1) the auxiliary verb *jesam* (to be) (the clitic form of the first person singular of the present), (2) the every person pronoun (oneself), or (3) an adjective meaning (alone, separated from others, without anyone else, lonely). The form appeared 31 times: 26 times as a form of the verb *jesam*, 4 times as a form of the pronoun, and once as a form of the adjective. The forms of the verb *jesam* were annotated correctly by spaCy and Stanza-I, but others missed 1–4 occurrences. Only spaCy annotated all pronouns , and TreeTagger and Stanza missed them all.

The form *tu* can be the demonstrative pronoun *taj* (this) (feminine singular in the accusative) (2 times) or the adverb *tu* (here) (6 times). Both taggers correctly annotated the occurrences of the pronoun, but Spacy was much better at annotating the adverb: it correctly tagged 5 out of 6 cases, while TreeTagger correctly tagged only one case.

References

1. Abney, S. Part-of-Speech Tagging and Partial Parsing. In *Corpus-Based Methods in Language and Speech Processing*; Young, S., Bloothooft, G., Eds.; Springer: Dordrecht, The Netherlands, 1997; pp. 118–136. [CrossRef]
2. Brill, E. A Simple Rule-Based Part of Speech Tagger. In Proceedings of the Speech and Natural Language: Proceedings of a Workshop, Harriman, NY, USA, 23–26 February 1992; pp. 112–116.
3. Giménez, J.; Màrquez, L. SVMTool: A general POS Tagger Generator Based on Support Vector Machines. In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04), Lisbon, Portugal, 26–28 May 2004; European Language Resources Association (ELRA): Lisbon, Portugal, 2004; pp. 43–46.
4. Wróbel, K. KRNNT : Polish recurrent neural network tagger. In *Human Language Technologies as a Challenge for Computer Science and Linguistics: 8th Language & Technology Conference, Poznań, Poland, 17–19 November 2017*; Vetulani, Z., Paroubek, P., Eds.; Fundacja Uniwersytetu im. Adama Mickiewicza: Poznań, Poland, 2017; pp. 386–391.
5. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv* **2015**, arXiv:1508.01991.
6. Schmid, H. Improvements in Part-of-Speech Tagging with an Application to German. In *Natural Language Processing Using Very Large Corpora*; Armstrong, S., Church, K., Isabelle, P., Manzi, S., Tzoukermann, E., Yarowsky, D., Eds.; Springer: Dordrecht, The Netherlands, 1999; pp. 13–25. [CrossRef]
7. Utvić, M. Annotating the Corpus of Contemporary Serbian. *Infotheca—J. Digit. Humanit.* **2011**, *12*, 36a–47a.
8. Vitas, D.; Krstev, C. Processing of Corpora of Serbian Using Electronic Dictionaries. *Prace Filologiczne* **2012**, *LXIII*, 279–292.
9. Honnibal, M.; Montani, I.; Honnibal, M.; Peters, H.; Landeghem, S.V.; Samsonov, M.; Geovedi, J.; Regan, J.; Orosz, G.; Kristiansen, S.L.; et al. Explosion/spaCy: v2.1.7: Improved Evaluation, Better Language Factories and Bug Fixes. 2019. Available online: https://zenodo.org/record/3358113 (accessed on 1 February 2022).
10. Šandrih, B.; Krstev, C.; Stanković, R. Development and Evaluation of Three Named Entity Recognition Systems for Serbian—The Case of Personal Names. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), Varna, Bulgaria, 2–4 September 2019; INCOMA Ltd.: Varna, Bulgaria, 2019; pp. 1060–1068. [CrossRef]
11. Bird Steven, E.L.; Klein, E. *Natural Language Processing with Python*; O'Reilly Media Inc.: Sebastopol, CA, USA: 2009.
12. Milovanović, B.; Stanković, R. Part of Speech Tagging for Serbian language using Natural Language Toolkit. In Proceedings of the 7th International Conference on Electrical, Electronic and Computing Engineering IcETRAN 2020, Etno village Stanišić, Bosnia and Herzegovina, 8–10 June 2020; Popović, D., Ed.; ETRAN Society: Belgrade, Serbia; Academic Mind: Belgrade, Serbia, 2020; pp. AII 1.1.1–AII 1.1.5.
13. Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; Manning, C.D. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Online, 5–10 July 2020; p. 8.
14. Ljubešić, N.; Dobrovoljc, K. What does Neural Bring? Analysing Improvements in Morphosyntactic Annotation and Lemmatisation of Slovenian, Croatian and Serbian. In Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing, Florence, Italy, 2 August 2019; Association for Computational Linguistics: Florence, Italy, 2019; pp. 29–34. [CrossRef]
15. Popović, Z. Taggers Applied on Texts in Serbian. *Infotheca—J. Digit. Humanit.* **2010**, *11*, 21a–38a.
16. Brants, T. TnT—A Statistical Part-of-Speech Tagger. In Proceedings of the Sixth Applied Natural Language Processing Conference, Seattle, WA, USA, 29 April–4 May 2000; Association for Computational Linguistics: Seattle, WA, USA, 2000; pp. 224–231. [CrossRef]
17. Sébastien, P. Unitex 3.0 User Manual. 2011. Available online: https://unitexgramlab.org/releases/3.1/man/Unitex-GramLab-3.1-usermanual-en.pdf (accessed on 11 March 2022).
18. Utvić, M. Izgradnja Referentnog Korpusa Savremenog Srpskog Jezika. Ph.D. Thesis, University of Belgrade, Faculty of Philology, Belgrade, Serbia, 2014.

19. Stanković, R.; Šandrih, B.; Krstev, C.; Utvić, M.; Škorić, M. Machine Learning and Deep Neural Network-Based Lemmatization and Morphosyntactic Tagging for Serbian. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; European Language Resources Association: Marseille, France, 2020; pp. 3954–3962.

20. Sjöbergh, J. Combining POS-taggers for improved accuracy on Swedish text. In Proceedings of the 14th Nordic Conference of Computational Linguistics, NoDaLiDa 2003, Reykjavik, Iceland, 30–31 May 2003; Volume 2003, p. 8.

21. Henrich, V.; Reuter, T.; Loftsson, H. CombiTagger: A System for Developing Combined Taggers. In Proceedings of the Twenty-Second International Florida Artificial Intelligence Research Society Conference, Sanibel Island, FL, USA, 19–21 May 2009; Lane, H.C., Guesgen, H.W., Eds.; AAAI Press: Palo Alto, CA, USA, 2009.

22. Aliwy, A.H. Combining POS taggers in master-slaves technique for highly inflected languages as Arabic. In Proceedings of the 2015 International Conference on Cognitive Computing and Information Processing(CCIP) 2015, Noida, India, 3–4 March 2015; pp. 1–5. [CrossRef]

23. Krstev, C. *Processing of Serbian—Automata, Texts and Electronic Dictionaries*; University of Belgrade, Faculty of Philology: Belgrade, Serbia, 2008; p. 228.

24. Ranka, S.; Cvetana, K.; Biljana, L.; Mihailo, Š. Electronic dictionaries-from file system to lemon based lexical database. In Proceedings of the 11th International Conference on Language Resources and Evaluation-W23 6th Workshop on Linked Data in Linguistics: Towards Linguistic Data Science (LDL-2018), LREC 2018, Miyazaki, Japan, 7–12 May 2018; pp. 48–56.

25. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.