Joint Ensemble Model for POS Tagging and Dependency Parsing

Iliana Simova **Dimitar Vasilev** **Alexander Popov** **Kiril Simov** **Petya Osenova**
Linguistic Modelling Laboratory, IICT-BAS
Sofia, Bulgaria
`{iliana|dvasilev|alex.popov|kivs|petya}@bultreebank.org`

Abstract

In this paper we present several approaches towards constructing joint ensemble models for morphosyntactic tagging and dependency parsing for a morphologically rich language – Bulgarian. In our experiments we use state-of-the-art taggers and dependency parsers to obtain an extended version of the treebank for Bulgarian, BulTreeBank, which, in addition to the standard CoNLL fields, contains predicted morphosyntactic tags and dependency arcs for each word. In order to select the most suitable tag and arc from the proposed ones, we use several ensemble techniques, the result of which is a valid dependency tree. Most of these approaches show improvement over the results achieved individually by the tools for tagging and parsing.

1 Introduction

Language processing pipelines are the standard means for preprocessing natural language text for various natural language processing (NLP) tasks. A typical pipeline applies the following modules sequentially: a tokenizer, a part-of-speech (POS) tagger, a lemmatizer, and a parser. The main drawback of such an architecture is that the erroneous output of one module in the pipeline propagates through to its final step. This usually has a more significant impact on the processing of languages with segmentation issues, like Chinese, or languages with rich morphological systems, like the Slavic and Romance ones, which exhibit greater morphological and syntactic ambiguity due to the high number of word forms and freer word order.

In this paper we present several experiments in which we simultaneously solve two of the aforementioned tasks – tagging and parsing. The motivation behind this idea is that the two tasks are highly dependent on each other when working with a morphologically rich language, and thus a better solution could be found for each if they are solved jointly. We assemble the outputs of three morphosyntactic taggers (POS taggers) and five dependency parsers in a single step. The ensemble approach uses weights in order to select the best solution from a number of alternatives. We follow (Surdeanu and Manning, 2010) and use two classes of approaches for selecting weights for the alternatives: *voting*, where the weights are assigned by simple calculations over the number of used models and their performance measures; *machine learning weighting*[1], where machine learning is exploited in order to rank the alternatives on the basis of a joint feature model. We refer to both types of approaches as *ranking*. The language of choice in our experiments is Bulgarian, but the techniques presented here are easily applicable to other languages, given the availability of training data.

The interaction between the two levels – morphology and syntax – is carried out via a joint model of features for machine learning. Its aim is to determine the best possible combination out of the predictions of the different taggers and dependency parsers. Working only with the outputs of the taggers and parsers, instead of considering all possibilities for tag, head and syntactic relation for each word in the sentence, reduces the search space and allows us to experiment with more complex features. One limitation of this approach is that the correct combination of an POS tag and a dependency arc might not have been

[1]Surdeanu and Manning (Surdeanu and Manning, 2010) call them *meta-classification.*

predicted by any of the tools in the first place. Therefore the ensemble approach can be beneficial only to a certain extent.

The data used throughout our experiments consists of the dependency conversion[2] of the HPSG-based Treebank of Bulgarian – the BulTreeBank. This data set contains non-projective dependency trees, which are more suitable for describing the relatively free word order of Bulgarian sentences.

The structure of the paper is as follows: in Section 2 we introduce related work on joint models and ensemble models; in Section 3 we introduce related work on Bulgarian parsing and POS tagging; in Section 4 we present our ensemble model; in Section 5 we report on our current experimental setup, including the construction of a parsebank of parses and tagging results; Section 6 presents the results from our ensemble experiments; the last section concludes the paper.

2 Related Work

Our work on ensemble systems for dependency parsing is inspired by the in-depth performance analysis of two of the most influential dependency parsing models: transition-based and graph-based (McDonald and Nivre, 2007). This analysis shows that the two frameworks make different errors when trained and tested on the same datasets. The authors conclude the paper by proposing three approaches for using the advantages of both frameworks: (1) ensemble systems – weighted combinations of the output of both systems; (2) hybrid systems – a single system designed to integrate the strengths of the individual ones; and (3) novel approaches – based on a combination of new training and inference methods. In their further work (Nivre and McDonald, 2008) on the subject they present a hybrid system that combines the two models. The work presented in this paper is along the lines of their first suggestion – a system to facilitate the combination of the outputs of several parsing and tagging models, in order to find an optimal solution.

An experiment with ensemble systems is presented in (Surdeanu and Manning, 2010). This work describes several approaches to the combination of dependency parsers via different types of voting and meta-classification. Voting determines the correct dependency arcs by choosing the ones that are selected by the majority of parsers. Weighted voting uses the accuracy of each parser in order to choose between their predictions for each arc. We also employ these two ranking techniques in our current experiment. Surdeanu and Manning (Surdeanu and Manning, 2010) conclude that meta-classification does not improve the results in comparison to voting. They divide the dependencies in two categories: majority dependencies and minority dependencies. Their conclusion is that meta-classification cannot provide a better selection of minority dependencies, and in this way is comparable to voting. In our work we show that depending on the feature selection for meta-classification, it can actually outperform the voting approach. The experiments presented in (Surdeanu and Manning, 2010) do not use a specific algorithm for the selection of dependencies, and do not ensure that the result of voting is a well-formed dependency tree. In our work we use two algorithms to ensure the construction of trees. We show that the results also depend on the algorithm for tree construction.

Joint models have been successfully used for processing other morphologically rich languages. For instance, (Lee et al., 2011) propose a joint model for inference of morphological properties and syntactic structures, which outperforms a standard pipelined solution when tested on highly-inflected languages such as Latin, Czech, Ancient Greek and Hungarian. It uses a graphical model that employs "local" and "link" factors to impose local word context constraints and to handle long-distance dependencies.

(Cohen and Smith, 2007) and (Goldberg and Tsarfaty, 2008) focus on a joint model for morphological segmentation and syntactic parsing with application to Hebrew. The authors argue that syntactic context is crucial for the correct segmentation of tokens into lexemes and that a model wherein the segmentation and parsing modules share information during processing is better suited to carry out the task. To solve the two tasks jointly, the different morphological analyses of a given utterance are represented simultaneously in a lattice structure; a path through the lattice corresponds to a specific morphological segmentation of the utterance. In (Cohen and Smith, 2007), paths in the lattice and parse trees are combined through a joint probability model and the best combination is found through chart parsing.

[2]www.bultreebank.org/dpbtb/

(Hatori et al., 2012) employ an incremental joint approach to solve three tasks in Chinese: word segmentation, POS tagging, and dependency parsing. The motivation for solving them simultaneously is that some segmentation ambiguities in the language cannot be resolved without considering the surrounding grammatical constructions, while syntactic information can improve the segmentation of out-of-vocabulary words. Parsing is done through a dynamic programming framework – a version of the shift-reduce algorithm.

Joint morphological and syntactic analysis of several morphologically rich languages is presented in (Bohnet et al., 2013). They use an extended transition system for dependency parsing to incorporate POS tagging, tagging with morphological descriptions and lemmas. In addition they define new evaluation metrics. They include the standard POS accuracy, Labeled and Unlabled Arc Accuracy, but also accuracy of combination of features like POS tags, morphological description, lemmas and dependency arcs. Several experiments with different parameters controlling the selection of best tags and morphosyntactic descriptions are presented.

The approach presented in our work is joint in the sense that we solve two tasks simultaneously – the choice for POS tag is dependent on the choice for dependency arc, and vice versa. However, our approach is also ensemble, since it combines the outputs of several systems for solving the two tasks, instead of exploring the whole search space of all combinations of tags and arcs. In this way, the approach is better described as a *joint ensemble model*.

3 Related Work on Bulgarian

Bulgarian is still under-studied with respect to parsing. Although several systems were trained on the BulTreeBank treebank during the CoNLL-X 2006 Shared Task (Buchholz and Marsi, 2006) and after it, a pipeline including a dependency parser with state-of-the-art performance does not exist. A state-of-the-art POS tagger with nearly 98% accuracy is available for Bulgarian (Georgiev et al., 2012). The best result for dependency parsing of Bulgarian reported in literature is 93.5% UAS (Martins et al., 2011). The best result for a pipeline including POS tagging and dependency parsing for Bulgarian is not known because most available tools were trained on the whole BulTreeBank and there is no way to measure their actual performance.

Our work is motivated by previous efforts to solve several NLP tasks simultaneously with application to Bulgarian (Zhikov et al., 2013). The presented joint model for POS tagging, dependency parsing, and co-reference resolution achieved results comparable to a state-of-the-art pipeline with respect to dependency parsing. This pipeline, however, used gold standard POS tags as input to the parser. Note that in the current work we do not rely on gold standard POS tags in the dependency parsing step in order to achieve more realistic results.

The usefulness of the ensemble parsing approach for Bulgarian is investigated in our previous works – (Simov et al., 2013) and (Simov et al., 2014). We trained 21 dependency parsing models with different configurations (parsing algorithm settings and features), including 12 MaltParser (Nivre et al., 2006) models, 6 MSTParser (McDonald, 2006) models, two TurboParser (Martins et al., 2010) models, and one Mate-tools parser (Bohnet, 2010) model. The best achieved ensemble result was 93,63% UAS, but gold POS tags were used as input for parsing. In our current work the best result is slightly lower, but more realistic, since no gold POS tags were used.

In this work as well as in the previous mentioned works we use Chu-Liu-Edmonds algorithm for maximum spanning tree as implemented in the MSTParser to ensure the construction of complete dependency trees. In (Zhikov et al., 2013), POS tags and the co-referential chains are encoded as an extension of the dependency tree in order to apply the same algorithm. We make use of this representation in the current work, as described in the following lines.

4 Ensemble Model

Our ensemble model works over extended dependency trees and graphs. First we define a dependency tree. Then we extend the dependency tree to include alternative POS tags for each wordform node and alternative dependency arcs.

Let us have a set D of dependency tags ($ROOT \in D$) and a sentence $x = w_1, ..., w_n$. A *dependency tree* is a tree $T = (V, A, \delta)$ where:

1. $V = \{0, 1, ..., n\}$ is an ordered set of nodes, that corresponds to an enumeration of the words in the sentence (the root of the tree has index 0);
2. $A \subseteq V \times V$ is a set of arcs;
3. $\delta : A \to D$ is a labeling function for arcs;
4. 0 is the root of the tree.

In order to extend the tree, we assume a range of possible POS tags for each wordform in the sentence. Such a range of tags has to contain the correct tag for the wordform in the given context. In this work we assume they are coming from results of several POS taggers. These tags are included in the tree as service nodes. In the linear representation of the sentence, they are inserted after the node for the corresponding wordform, and before the node for the next wordform to the right. They are connected to the corresponding wordform with a special link $TAG. In order to indicate the correct tag, we introduce another type of service node. In the linear representation of the sentence, it is inserted after the last POS tag candidate node, and before the one corresponding to the next wordform to the right. This node is connected to the correct tag via a special arc $CTAG (correct tag). In this way, all information about the potential tags and the correct tag is represented in the form of a subtree, attached to the wordform. Figure 1 depicts the encoding of a word with POS tag ambiguity as a tree. The correct tag is indicated: verb, personal, perfective, transitive, finite, aorist, third person, singular, or "Vpptf-03s". The `TAG` arcs are represented as red links. The `CTAG` arc is represented as an oval.

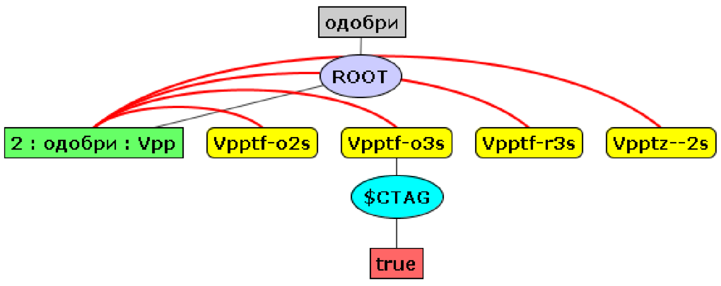

Figure 1: Subtree of a word with candidate POS tags and the correct tag.

Our ensemble model starts working on a set of extended dependency trees from which it to select the an extended tree. We represent this set as an extended dependency graph.

Let us have a set G of POS tags, and a set D of dependency tags ($ROOT \in D$). Let us have a sentence $x = w_1, ..., w_n$. An *extended dependency graph* is a directed graph $Gr = (V_e, A, \pi, \delta, \rho)$ where:

1. $V_e = \{0, 1, \$TAG1_1, TAG1_2, ..., TAG1_{j_1}, TT1, ..., n, TAGn_1, TAGn_2, ..., TAGn_{j_n}, TTn\}$ is an ordered set of nodes, that corresponds to an enumeration of the words in the sentence (the root of the tree has index 0), additional nodes for alternative POS tags - $TAGk_j$, such that for each wordform k there is at least one such node and for each wordform k there is one TTk selecting its correct POS tag;
2. $V = \{0, 1, ..., n\}$ is the ordered subset of V_e corresponding to words in the sentence including the root element;
3. $A \subseteq V \times V$ is a set of arcs;
4. $\pi : TAGk_j \to G$ is a labeling function from tag nodes to POS tags (node 0 does not have POS tag). These nodes are called POS nodes;
5. $TAGk_j$ is connected by an arc with label $TAG to the wordform k;
6. TTk is connected each $TAGk_j$ by an arc with label $CTAG, where k is the number of the wordform;

7. $\delta : A \rightarrow D$ is a labeling relation for arcs. Each arc has at least one label;
8. $\rho : \langle a, l \rangle \rightarrow R$, is a ranking function, where a is either an arc in A and $l \in \delta(a)$ or $a = \langle TAGk_j, k \rangle$ and $l = \$CTAG$. ρ assigns to each labeled dependency arc or tagging arc a rank. The arcs from POS tags nodes to wordform node always have rank 1;
9. 0 is the root of the graph.

We use an extended dependency graph *Gr* to represent the initial data from which we select an analysis. Each extended dependency graph could incorporate the results from several POS taggers and several dependency parsers. At the beginning each node for true tag is connected to all corresponding tagger predictions, and after ensemble is assigned to a single parent node, the correct tag. In this way, all information about the potential tags and the correct tag is represented in the form of a subtree, attached to the word form.

Our ensemble model starts from an extended dependency graph *Gr* and constructed an extended tagged dependency tree *T* which is a subgraph of *Gr*. In the rest of the paper we will use just dependency tree and dependency graph terms to denote the extended ones. We use two algorithms for the construction of a single dependency tree from the predictions of all tagger and parser models (dependency graph).

The first algorithm, denoted `LocTr`, is presented in (Attardi and Dell'Orletta, 2009). It constructs the dependency tree incrementally, starting from an empty tree and then selecting the arc with the highest rank that could extend the current partial tree. The arcs are selected from the extended dependency graph. The algorithm chooses the best arc *locally*.

The second algorithm, denoted `GloTr`, is the Chu-Liu-Edmonds algorithm for maximal spanning tree implemented in the MSTParser (McDonald, 2006). This algorithm starts with a complete dependency graph including all possible dependency arcs. Then it selects the maximal spanning tree on the basis of the ranks assigned to the potential arcs. The arcs that are not proposed by any of the parsers are deleted (or we could think about them as having infinite small rank). The arcs for the service nodes include only the once from the definition of extended dependency graph. The algorithm is *global* with respect to the selection of arcs. In our scenario, however, we do not construct a full graph, but one containing only the suggestions of the parsers and taggers.

These two ensemble algorithms are included in our system for experiments with dependency parsers. The user can specify which one should be used in their experiments, or alternatively compare the performance of both. Making choice for each of the TT nodes both algorithms select the best POS tag for the corresponding wordform.

In the next section we define the experimental setup: the creation of parsebank where for each tree in the original treebank a dependency graph is created; the definition of voting approaches and the machine learning weighting.

5 Experimental Setup

In this section we present in detail the way in which our ensemble experiment was set up, including the data format and choice of ranking and features for machine learning.

5.1 Tagger and Parser Models and Parsebank

In the current experiments we use the five parsing models which achieved the highest LAS and UAS scores for the BulTreeBank data in previous experiments[3] (Simov et al., 2014). They include two MaltParser models, `MLT07` and `MLT09`, one MSTParser model, `MST05`, one TurboParser model, `Turbo02`, and one Mate-tools Parser model, `MATE01`. The following configurations were used for each model:

1. `MLT07` - Convington non-projective algorithm with extended feature set for lemmas.
2. `MLT09` - Stack eager algorithm with extended feature set for morphosyntactic descriptions.
3. `MST05` - default parser settings, with the exception of the order of features. The parser was set to use features over pairs of adjacent edges (*second-order*: true).

[3]The names of the models were left unchanged for easier reference to previous work.

4. `MATE01` - default parser settings.
5. `Turbo02` - the parser was set to use a complex set of features (*model_type*=full), which include arbitrary sibling parts, non-projectivity parts, grand-sibling third-order parts, and tri-sibling third-order parts.

The models were initially trained on the gold standard values for lemma, part-of-speech tags and features, from the dependency version of the BulTreeBank. The best performing model in a 10-fold cross validation is `MATE01`, with 92.9% UAS, followed by `TURBO02` with 92.7% UAS.

In addition to the parsing models, three part-of-speech taggers were trained to predict the morphosyntactic tags from the BTB-tagset (Simov et al., 2004) for the data. They include a baseline tagger which makes use of a lexicon (BLL tagger), the morphology tagger of Mate-tools, and the TreeTagger (Schmid, 1994).

The standard approach for setting up a POS tagging baseline – selection of the most frequent tag – cannot be applied for our experiment with Bulgarian, because of the rich morphosyntactic tagset of 680 tags. We construct a baseline tagger on the basis of the corpus and a morphological lexicon. This baseline ignores context altogether and assigns each word type the POS tag it was most frequently seen with in the training dataset; ties are broken randomly. For words not seen in the training dataset we use a simple guesser which assigns POS tags on the basis of the word suffix. We first built two frequency lists, containing respectively (1) the most frequent tag in the training dataset for each word type, as before, and (2) the most frequent tag in the training dataset for each class of tags that can be assigned to some word type, according to the lexicon. Given a target word type, this new baseline first tries to assign to it the most frequent tag from the first list. If this is not possible, which happens (i) in case of ties or (ii) when the word type was not seen during training, it extracts the tag class from the lexicon and consults the second list. If there is a single most frequent tag in the corpus for this tag class, it is assigned; otherwise a random tag from this tag class is selected. This strategy gives us a very high accuracy for this tagger. Although we refer to it as a baseline, it achieves the best score among the taggers we used in these experiments. Our explanation for this is the fact that we are using a morphological lexicon to predict the possible tags for the unseen words in the test sets.

The Mate morphology tagger constitutes one step of the processing pipeline in Mate-tools, and makes use of previously predicted values for lemma and tag. Therefore, we trained a Mate-tools lemmatizer and tagger in addition, so that no gold standard data is used directly to obtain the morphological information for each word in our experiment.

The third tagger trained on the BulTreeBank data and used in the experiments is TreeTagger, a tool that estimates transition probabilities via decision trees. In training mode it was run with its default options. The tagger takes as parameters a lexicon of all the words that are found in the training corpus, one per line, followed by the respective POS tags encountered in the corpus and, optionally, by the lemmas for those forms. We extracted this information from our training data, but skipped the optional training for lemma, since lemmas can be automatically generated for each word form using the predicted BTB tag.

Using the taggers in a 10-fold experiment, we obtained three new versions of the dependency treebank, with predicted values for the fields lemma, tag, and features, which brings us closer to a real-world parsing scenario with unseen data. The output of the Mate morphology tagger is a prediction of the values in the features field of the CoNLL dependency format. The BLL and TheeTagger predictions for morphosyntactic tags were used to generate the fields lemma, tag, and features, for each word in the original treebank. The taggers achieved accuracy of 95.91% (BLL Tagger), 94.92% (Mate morphology tagger), and 93.12% (TreeTagger). Each of the five parsing models was evaluated on the new data sets (Table 1). This evaluation exemplifies the extent to which a decrease in tagger accuracy can influence parsing accuracy. There is a decrease in performance in terms of UAS score ranging from 1.6% to 4.6%, compared to the performance of the models when using the gold data fields.

We define an upper bound for the potential improvement through ensemble for each task as the percentage of words in the parsebank for which there is at least one correct prediction by a tagger or parser. The upper bound for the combination of taggers is 98.38%. For the parses the upper bound is 96.95 %

MLT07	MLT09	MATE01	MST05	Turbo02	training data
0.900	0.908	0.929	0.911	0.927	gold
0.881	0.890	0.910	0.890	0.911	BLL tagger
0.881	0.889	0.908	0.890	0.910	Mate tagger
0.857	0.865	0.883	0.865	0.883	TreeTagger

Table 1: Average UAS scores from the 10-fold cross validation of the parsing models trained on gold data and on data containing automatically generated fields obtained using the outputs of three taggers.

for UAS and 95.38 % for LAS. These upper bounds can be reached if the algorithm is able to select the correct solution in all cases.

A rich search space of possible combinations of POS tags and parses is available for the voting and machine learning weighting modules to choose from. An increase in tagging accuracy through ensemble can lead to obtaining better parsing results. In order to allow for ensemble to be performed on the output of several POS taggers and parsers, the tree that stores the POS tags and the head and relation predicted by each parsing model was represented in a dependency graph.

Thus, the parsebank consists of dependency graphs constructed by the three POS tagging models and the five dependency parsing models. All the models are trained on gold data from the original treebank. Because the parsing depends on the POS tags assigned to the wordform we applied the parsing models for the results from each POS taggers. In this way we potentially up to fifteen arcs per wordform.

5.2 Combining Parses by Voting

We investigate three voting modes for the calculation of the weight assigned to each candidate dependency arc: (1) the arcs are ranked by the number of parsers/taggers that predicted them (Rank01); (2) the arcs are ranked by the sum of the accuracy of all parsers/taggers that predicted them (these metrics include the LAS and UAS measures from the 10-fold cross validation and the tagger accuracies individually achieved by each tool) (Rank02); and (3) the arcs are ranked by the average of the accuracy of the parsers/taggers that predicted them (Rank03).

5.3 Combining Taggers and Parsers by Machine Learning Weighting

In order to evaluate the interaction between morphosyntactic information and the dependency parsing, we conducted an experiment in which a machine learning technique was used for ranking the tags and arcs suggested by the different models. This was done with the help of the package `RandomForest`[4] of the system R[5]. The parsebank was once again divided into training and test parts, using the same proportion, but orthogonally: 90% and 10%.

For each word node there are up to three different tags and for each tag there are up to five arcs. We constructed pairs of tags and arcs on the basis of these suggestions. Each pair was compared with the gold data and classified as correct or incorrect for a given context. To each pair (`Tag` , `Arc`) a vector of features was assigned. `Arc` was modelled by three features: relation (`Rel`), distance in words to the parent node (`Dist`) and direction of the parent node (`Dir`) – `Left`, meaning that the parent node is on the left, and `Right`, meaning that the parent node is on the right. `Tag` was represented as a vector of its grammatical features including POS, gender, number, etc. In this way the agreement features were represented explicitly. We also included the word form string as a feature, as well as the corresponding information for the word form context – words before and after it, and the same for the parent node in the dependency tree.

A representation of this data as a value vector for `RandomForest` is given in Table 2.

[4]http://cran.r-project.org/web/packages/randomForest/randomForest.pdf
[5]http://www.r-project.org/

Feature	Value
`Word`	the current node
`WordBefore`	the word before the current node
`WordAfter`	the word after the current node
`ParentWord`	the parent word
`PWordBefore`	the word before the parent word
`PWordAfter`	the word after the parent word
`SelectedArc`	one of the arcs suggested by one of the models for the node
`SelectedTag`	one of the arcs suggested by one of the models for the node
`CorrectIncorrect`	`true` or `false` depending on whether the selected pair is the correct one for the node

Table 2: Feature vector used with RandomForest for the experiment.

The tuples generated from the training part of the treebank were used to train the `RandomForest` in regression mode, then the model was applied to the test set to rank each pair. After this the ranks were distributed to tags and arcs. These weights were used by the algorithms `LocTr` and `GloTr`.

Each tag and arc for a given word could participate in several different feature vectors. Thus each of them could receive different weights from the evaluation of the vectors. In our view, the best selection among these weights could be determined only through experimentation. We have tested three rankings: WMax – the maximum tag weight for all word vectors, WMin – the minimum tag weight for all word vectors, and MSum – the sum of all tag weights for all word vectors.

6 Experiments

We ran both algorithms (`LocTr` and `GloTr`) for construction of dependency trees using various combinations of the outputs of our dependency parsing and tagging models. Table 3 shows the parsing accuracy results when combining all models (1), only the models of the two best performing parsers, Turbo02 and Malt01 (2), and the best combination we have found by trying all possible combinations (around 32K) (3). We included the results for (1) and (2) to demonstrate that the best combination cannot be predicted in advance by simply selecting the candidate with the largest number of models, or the one with the best performing individual parsers.

The best combination in this experiment in terms of UAS score is: `MLT09+BLL`, `Mate01+BLL`, `MST05+BLL`, `Turbo02+BLL`, `MLT07+MateTagger`, `Mate01+MateTagger`, `Turbo02+MateTagger`. This combination achieves better UAS score (92.47%) than any of the individual parsers (see Table 1).

There was an improvement of 1.37% over the best performing individual parser `Turbo01+BLL`, which achieves 91.10% UAS. The unlabelled accuracy after voting is, however, still 0.43% lower than the best result on the gold data achieved by an individual model `Mate01`. We suspect that this is due to having only three tagger models in the current experiment, and that adding a few more tagger models for voting can help improve the result.

Table 4 presents the accuracy achieved for all possible combinations of the three taggers by voting per rank. We have to stress the fact that the selection of the morphosyntactic tag in the extended dependency tree is independent from the selection of dependency arcs, because each new tag node is connected to the word node by equal weight. The interaction between dependency arcs and morphosyntactic arcs is ensured by the features used in machine learning weighting. The results for the combination improve the individual accuracy for all taggers.

The results in Table 4 show that it is hard to predict the best combinations in advance without enumerating all possibilities. Note that for voting (Rank01, Rank02, and Rank03) it is meaningless to investigate

Models		Algorithm	Rank01		Rank02		Rank03	
			Number		Sum		Average	
			LAS	UAS	LAS	UAS	LAS	UAS
(1)	all models	`LocTr`	88.55	92.05	88.61	92.10	85.57	88.82
		`GloTr`	88.55	91.96	88.65	92.04	84.54	88.75
(2)	all `Mate01` and `Turbo02` models	`LocTr`	87.68	91.38	87.80	91.48	86.94	90.55
		`GloTr`	87.58	91.21	87.82	91.45	86.84	90.62
(3)	best combination	`LocTr`	88.90	92.34	89.05	**92.47**	86.19	89.40
		`GloTr`	88.94	92.31	89.14	**92.45**	85.23	89.27

Table 3: UAS and LAS obtained after voting using the algorithms `LocTr` and `GloTr` for tree construction. (1) All 18 models; (2) A combination of the best individual models: `Mate01` and `Turbo02` + each tagger; (3) best combination: `MLT09+BLL`, `Mate01+BLL`, `MST05+BLL`, `Turbo02+BLL`, `MLT07+MateTagger`, `Mate01+MateTagger`, `Turbo02+MateTagger`;

Voting	Rank01	Rank02	Rank03
	Number	Sum	Average
`BLL, Mate, TreeTagger`	96.24	96.24	95.22
MLearning	**WMax**	**WMin**	**WSum**
`BLL, Mate, TreeTagger`	96.10	96.20	96.25
`BLL, Mate`	96.62	96.59	**96.63**
`BLL, TreeTagger`	95.89	96.08	96.09
`Mate, TreeTagger`	95.29	95.40	96.25

Table 4: Tagger accuracy after voting and machine learning weighting.

the combinations involving only two taggers, because in this case the output of voting will always be the same as the output of the better tagger.

Table 5 presents the UAS and LAS measures achieved using machine learning weighting. In this case the best combination is `Mate01+BLL`, `Turbo02+BLL`, `Mate01+MateTagger`, `Turbo02+MateTagger`. Again, the results are better than the ones obtained by the individual parsing models. They also demonstrate some small improvement over the voting ranking.

Model	Algorithm	LAS	UAS
all	`LocTr`	89.17	92.46
	`GloTr`	89.23	92.27
all `Mate01` and `Turbo02` models	`LocTr`	88.26	91.81
	`GloTr`	88.32	91.87
best combination	`LocTr`	89.76	93.18
	`GloTr`	89.81	93.22

Table 5: Results from the experiments with `RandomForest`. The best combination is `Mate01+BLL`, `Turbo02+BLL`, `Mate01+MateTagger`, `Turbo02+MateTagger`.

These experiments show the following: (1) the combination of taggers and parsers is a feasible task; (2) the combination improves the accuracy of both the taggers and the parsers; (3) the combination of both tasks is better than the pipeline approach; (4) there is room for improvement in order to reach the upper bounds presented in Section 5.1.

7 Conclusion and Future Work

In this paper we have presented several approaches for combining parses produced by five parsing models and tagging results from three taggers. The motivation behind a joint ensemble model is the interaction between the morphosyntactic features of the word forms and the dependency relations between them. The interaction could be considered as local and global interaction. The local interaction is usually captured by n-gram models for tagging. The global interaction is represented by such phenomena like subject – verb agreement, verb clitic – object – indirect object agreement, agreement between head noun and relative pronouns, agreement between secondary predication, agreement within co-reference chains, agreement within NPs. With relation to these cases, our current model deals with local interaction on the basis of an n-gram model. Global agreement phenomena are currently modeled via dependency arcs between word forms that agree in their grammatical features.

We deal with some of the interaction between local and some global patterns via a machine learning approach in which the appropriateness of the MorphoSyntactic tag and the dependency arc for a given word form are evaluated in conjunction. The appropriateness is expressed as a number between 0 and 1, where 0 means inappropriate and 1 means appropriate. This number is used as a rank for the ensemble algorithms.

Some of the global agreement phenomena such as verb clitic – object – indirect object agreement, secondary predication and relative pronoun agreement are not covered by the current model. In the future we plan to extend the model with global features defined not by arcs in the dependency tree, but by patterns of dependency paths. These feature patterns will depend on the grammatical characteristics of the given word form. In some cases they might not be directly related to the word form in the tree.

Our experiments show that a joint architecture is a good alternative to a pipeline architecture. There is an improvement in accuracy for both tasks in our joint model. However, this approach has its limitations with respect to possible improvement.

Future extensions of the experiments in several directions are envisaged. First, more linguistic knowledge will be included from the morphological lexicon, valency lexicon and semantic categories of the words as features for machine learning. Second, we plan to extend the experiments by including more tagger and parser models, which could lead to an increase in the upper bound for potential improvement in accuracy. In future work we envisage to compare our work with the work of (Bohnet et al., 2013) applied on Bulgarian data. Also we will would like to include as features word clusters as they suggested in the paper and as we did in parsing context (Ghayoomi et al., 2014).

Acknowledgements

This research has received partial funding from the EC's FP7 (FP7/2007-2013) under grant agreement number 610516: "QTLeap: Quality Translation by Deep Language Engineering Approaches" and grant agreement number 611760: "EUCases: EUropean and National CASE Law and Legislation Linked in Open Data Stack".

References

Giuseppe Attardi and Felice Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 261–264, Boulder, Colorado.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, RichÃ¡rd Farkas, Filip Ginter, and Jan Hajic. 2013. Joint morphological and syntactic analysis for richly inflected languages. *TACL*, 1:415–428.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 89–97, Stroudsburg, PA, USA.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City.

Shay B Cohen and Noah A Smith. 2007. Joint morphological and syntactic disambiguation. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Prague, Czech Republic.

Georgi Georgiev, Valentin Zhikov, Kiril Ivanov Simov, Petya Osenova, and Preslav Nakov. 2012. Feature-rich part-of-speech tagging for morphologically complex languages: Application to Bulgarian. In *EACL'12*, pages 492–502.

Masood Ghayoomi, Kiril Simov, and Petya Osenova. 2014. Constituency parsing of bulgarian: Word- vs class-based parsing. Proceedings of LREC 2014.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *ACL 2008*, pages 371–379.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1045–1053.

John Lee, Jason Naradowsky, and David A Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 885–894.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 34–44, Stroudsburg, PA, USA.

Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 238–249, Edinburgh, Scotland, UK.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: a data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of international conference on new methods in language processing*, volume 12, pages 44–49. Manchester, UK.

Kiril Simov, Petya Osenova, and Milena Slavcheva. 2004. BTB:TR03: BulTreeBank morphosyntactic tagset BTB-TS version 2.0.

Kiril Simov, Ginka Ivanova, Maria Mateva, and Petya Osenova. 2013. Integration of dependency parsers for Bulgarian. In *The Twelfth Workshop on Treebanks and Linguistic Theories*, pages 145–156, Sofia, Bulgaria.

Kiril Simov, Iliana Simova, Ginka Ivanova, Maria Mateva, and Petya Osenova. 2014. A system for experiments with dependency parsers. In *Proceedings of LREC 2014)*, Reykjavik, Iceland.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of the North American Chapter of the Association for Computational Linguistics Conference (NAACL-2010)*, Los Angeles, CA.

Valentin Zhikov, Georgi Georgiev, Kiril Simov, and Petya Osenova. 2013. Combining pos tagging, dependency parsing and coreferential resolution for Bulgarian. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 755–762, Hissar, Bulgaria.