Introducing Shallow Syntactic Information within the Graph-based Dependency Parsing

Nikolay Paev, Kiril Simov, Petya Osenova
Artificial Intelligence and Language Technology
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Bulgaria
nikolay.paev@iict.bas.bg, kivs@bultreebank.org, petya@bultreebank.org

Abstract

The paper presents a new BERT model, fine-tuned for parsing of Bulgarian texts. This model is extended with a new neural network layer in order to incorporate shallow syntactic information during the training phase. The results show statistically significant improvement over the baseline. Thus, the addition of syntactic knowledge - even partial - makes the model better. Also, some error analysis has been conducted on the results from the parsers. Although the architecture has been designed and tested for Bulgarian, it is also scalable for other languages. This scalability was shown here with some experiments and evaluation on an English treebank with a comparable size.

1 Introduction

In this paper we present a transformer-based architecture for dependency parsing which is extended to accommodate some predefined shallow dependency information. The predefined information came from two sources: lexicons and shallow grammars. The Dependency information — dependency relations (arcs and labels) — are represented within the lexicon at least in two varieties: (1) representation of valency frames, and (2) representation of multiword expressions (MWEs). For a recent overview see (Giouli and Barbu Mititelu, 2024). In our in-house lexicons we use partial dependency trees in order to represent the obligatory grammar information such as the object and clitic relations of the verbal head and the modification relations of the nominal head. For example, the MWEs "kick the bucket" is expected to have in the lexicon two dependency relations — from the article "the" to the head noun "'bucket' the relation is "`det`" and from "bucket" to the head verb "kick" the relation is "`obj`".

Our goal was set to implement a parser that is able to incorporate preliminary dependency relations among words — even partial — from the lexicon, thus before parsing of the whole sentence this step has been performed. Similarly, shallow grammars provide sets of rules over the word forms and their grammatical annotation. These grammars are known to produce partial but reliable analyses mostly achieving 100 % accuracy. In the experiments reported in this paper both sources have been explored. The experiments and evaluation were performed for Bulgarian and English.

The structure of the paper is as follows: the next section provides a focused overview of related work; section 3 describes the Dependency parsing architecture that was implemented in our work. This section also elaborates on the modification of the initial architecture towards the incorporation of some sure information from lexicons and shallow grammars. In section 4 the experimental settings are described in detail. Here also the results are presented and discussed. In section 5 the manual evaluation of the results is outlined. Section 6 concludes the paper and presents some future directions of research.

2 Related Work

Zhou et al. (2023) show that prepositional phrase attachment poses the biggest challenge to understanding syntax by LLMs. The case study on training the dynamics of the LLMs revealed that the majority of syntactic knowledge is learned during the initial stages of training. For these reasons, we started with the injection of partial but sure linguistic information into the model. Shen et al. (2021) propose a new syntax-aware language model — Syntactic Ordered Memory (SOM). The model explicitly models the structure with an incremental parser and maintains the conditional probability setting a standard language model (left-to-right). The related experiments show that SOM can achieve strong results in language modeling, incremental parsing and syntactic generalization

Proceedings of the 22nd Workshop on Treebanks and Linguistic Theories (TLT 2024), pages 46 - 54

tests, while using fewer parameters than other models. The model uses constituency trees for English and these trees are embedded in a grid-like memory representation. The authors report improvement on phenomena like gross syntactic states and long-distance dependencies. In our case instead of incremental approach we use predefined partial syntactic information. Yoshida et al. (2024) propose a novel method called tree-planting. This means to implicitly "plant" trees into attention weights of Transformer LMs to reflect syntactic structures of natural language. Transformer LMs trained with tree-planting are called Tree-Planted Transformers (TPT). They learn syntax on small treebanks via tree-planting and then scale on large text corpora via continuous learning with syntactic scaffolding. Our approach is similar since it uses dependency subtrees but it relies only on chunks and MWEs as 'islands of certainty'. Another difference is that we add syntactic information within the transformer network during the fine-tuning phase, but in future we plan to pre-train a model on partially annotated corpora.

The combination of information from different sources in order to improve the overall performance of the parser is not a new idea. This is especially true for the combination of various machine learning techniques with sure symbolic knowledge under the motto *"why to guess if we already know?"*. With respect to dependency parsing Özates et al. (2020) use special rules to introduce dependency relations between certain word forms in the sentences. Each rule identifies some arcs within the dependency tree. The rules are applied recursively up to the moment when no more applications are possible. The result from the application of the rules is encoded as additional token embeddings which are concatenated with embeddings used by the actual neural network parser. The parser used in their experiments is an LSTM-based dependency parser — Stanford's Graph-based Neural Dependency Parser (Dozat et al., 2017). The baseline is the parser trained without these extended embeddings, and thus later trained with them. The paper reports on improving UAS (about 2 %) and LAS (near 3 %). Our approach differs from theirs in several ways: (1) We fine-tune a BERT[1] language model as a dependency parsing model. The fine-tuning step requires the existence of a dependency treebank[2]. In addition to the treebanks we relied on the supplement of "suggested" arcs which would facilitate arcs prediction and labeling (see below). These arcs we considered to be the linguistic knowledge added to the respective treebank. (2) the linguistic knowledge added during the training is not necessary to be the same as during the inference time. In this way the approach could be used when there are no reliable sources of such linguistic knowledge. The existence of a treebank, of course, is obligatory. (3) The additions of dependency relations in parallel to the treebank look like redundant information, but it plays an important role during the parsing of new texts.

In the next section we present the specifics of the dependency parsing model that has been implemented for the experiments reported in this paper.

3 Graph-based Dependency Parsing

In the implementation of our dependency parsing based on LLMs we follow the approach of McDonald et al. (2006) about a graph-based dependency parsing performed in two steps: (1) determination of dependency arcs in the syntactic tree — the immediate domination relation over the tokens in the sentence — for each token to find its immediate parent token (adding special token for the root of the sentence); and (2) labeling the selected arcs with the appropriate dependency relations. This approach was adopted by many of the recent dependency parsers ((Dozat and Manning, 2017), for instance) — where a transformer-based model is used for determining the context-aware token embeddings, and an additional model for the selection of the arcs (Head selection model) as well as for the labels.

In our implementation both - the transformer model and the head selection model - are directly connected - the head selection model is integrated as an additional layer over the last layer of the transformer model. The head selection model is similar to any other token classification model, except that the number of classes is dynamic — the number of possible heads in the sentence varies. When sub-word tokenization is performed, only the first token of each word is used, while the others are ignored during training and inference phases. In the next sections the implementation of the parser is presented in more details.

[1]BERT model is introduced by Devlin et al. (2018).

[2]In our experiments the available Bulgarian and one of the English Universal Dependency Treebanks — https://universaldependencies.org/

3.1 Head Classification — (UAS)

As it was mentioned above, the first step of the parser is to identify the arcs. This is done by selecting the head of each word form in the sentence. The head could be any of the other word forms in the sentence, or a specially included token for the root of the sentence.

Since the number of the possible heads in a sentence is dynamic — (it depends on the number of tokens within the sentence) — a simple linear (affine) transformation is not applicable. Instead, a self-attention mechanism is used due to its ability to aggregate information from sequences with different lengths.

Let $s = (w_0, w_1, ..., w_S)$ denote a sentence of length S, where w_0 is the special token for the head of the root. The representation of w_0 within the transformer encoding of the sentence is associated with the [CLS] token. In order to use a technique similar to self-attention, we exploit some parts of the corresponding matrices for each word form in the sentence. Thus we define the following matrices and vectors:

- Let $h_i = Model(w_i)$ be the embedding of w_i, produced by an encoder model. ($h_i \in \mathbf{R}^{d_e}$) for $i \in [0, S]$;
- Let $q_i = QueryMatrix(h_i)$ and $k_j = KeyMatrix(h_j)$ ($q_i, k_j \in \mathbf{R}^{d_k}$) be linear (affine) transformations of h_i for $i \in [1, S]$ and of h_j for $j \in [0, S]$;
- Let $K \in \mathbf{R}^{d_k \times S+1}$ be the matrix with rows - k_j for $j \in [0, S]$.

The distribution over all possible heads of w_j is obtained with softmax across the multiplication of q_i and K ($q_i K \in \mathbf{R}^{S+1}$). The encoder model weights and the transformations are trained with a cross entropy loss between the distribution over the heads and the one-hot encoded label of the correct head:

$$\text{Loss} = -\sum_{i=1}^{S}\sum_{k=0}^{S} y_{i,k} \log((\text{softmax}(q_i K))_k)$$

where:

- S is the sequence length.
- $y_{i,k} = \begin{cases} 1, & \text{if } w_k \text{ is the head of } w_i \\ 0, & \text{otherwise} \end{cases}$
- $q_i \in \mathbf{R}^{d_k}$ is the output vector of the query matrix transformation for the word w_i.
- $K \in \mathbf{R}^{d_k \times S+1}$ is the matrix with rows - the outputs of the key matrix transformation for all words in the sentence.

This end-to-end training fine-tunes the encoder model weights. It is done simultaneously over all words in a sentence and over multiple sentences in a batch. During inference the model produces distribution over the possible heads for each word. A simple strategy to predict the head of each word is to calculate the $argmax$ of the distribution.

$$\text{Prediction}(w_i) = argmax_{k=0}^{S}(q_i K)_k$$

We use this prediction for validation during training. However, it is well known that this greedy prediction does not guarantee a construction of a tree (although in more than 95 % of the cases a tree is produced). Thus, we adopted the `Chu-Liu-Edmonds` algorithm for the construction of a Maximum Spanning Tree over the full graph of all potential dependency arcs of the sentence to implement and to select the most probable tree (McDonald (2006)). The full graph in our case is presented as a transition matrix composed of the vectors for the distribution of the possible heads for each word in the sentence. Over this graph we apply the `Chu-Liu-Edmonds` algorithm.

Our head classification layer can be seen as a simplified version of the (Dozat and Manning, 2017) Deep Bi-affine attention. While they use an LSTM to produce embeddings, we use BERT and our scores are just the dot products of the heads and dependents transformations while they transform the outputs and then use a Bi-affine transformation to produce the scores. We argue that a simpler layer is sufficient, because of the expressive power of the pre-trained BERT.

3.2 Incorporating the Information from the Lexicons and Shallow Grammars

As it was mentioned above, our goal is to incorporate "sure" information about the syntactic structure of a given sentence in the process of parsing in such a way that it improves the performance of the parser. Such information could be used during the training time (fine-tuning) of the parser as well as during the inference time.

Let $s = (w_0, w_1, ..., w_S)$ denote a sentence of length S, and w_0 is the special token for the root as above. T is a dependency tree for s if and only if

$$T = \{(w_i, w_j, l_i) | W_i \in s, w_j \in s, l_i \in DL\},$$

where T is a tree, the root of T is w_0, w_j is the head node of w_j, and DL is a set of labels for the dependency relations. The "sure" syntactic information for the sentence s is a subset of arcs in the dependency tree T: $T_P \subseteq T$. We call T_P a set of prompting arcs. In the experiments reported in this paper we use only unlabeled arcs, because we would like to see their influence on the unlabeled parsing. In our opinion, the additional information — the labels and grammatical features — might be incorporated in a similar manner. Thus, T_P contains arcs from some of the words in the sentence to their heads.

Our ***main intuition*** is that we could use the prompting arcs to urge the model to pay more attention to the "sure" heads provided by T_P.

The incorporation of the additional information from the set T_P can be done in at least two ways. First, through a simple extension over the model, described in the previous subsection, is to modify the scores for the corresponding arcs predicted by the model to an infinitely large score. In this way we will force the `Chu-Liu-Edmonds` algorithm to always select these arcs. A similar approach could be used with the argmax head selection algorithm.

One disadvantage of this method is that it has an effect only during the actual head selection phase. Thus, the transformer model cannot take advantage of the predefined arcs. Motivated by the intuition that incorporating the information as early as possible would facilitate the model predictions for the other words, as a second solution we propose an extension to the layers of the encoder model, which are used to prompt the model with the predefined information. Since only a fraction of the dependency arcs are predefined, this prompting is done only on a small number of words in the sentence. Thus, to implement this intuition within the model we modify the typical architecture of the transformer model.

Let us consider the standard architecture of the transformer block for the encoding model. It consists of two major elements: the first element includes a Multi-Headed Self-Attention layer (MHA) with following residual connections and normalization. The second element is a Feed-Forward Network (FFN), also with following residual connections and layer normalization (LN). The output of each of the elements are denoted in the following way:

$$O_{attn}(X) = LN(X + MHA(X))$$

is the result of the first element — Multi Headed Self-Attention, residual connections and layer normalization. Then

$$O_{ffn}(X) = LN(X + FFN(X))$$

is the result of the second element — Feed-Forward Network, residual connections and layer normalization.

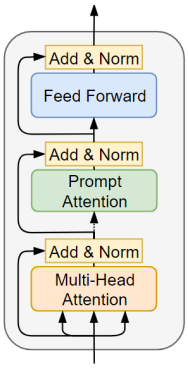

Figure 1: The modified encoder layer with prompt attention

Our modification introduces a bias to the embedding of each token towards the embedding of its predefined head. This is done by the prompting attention (PA) sublayer:

$$O_{pa}(X) = \left(LN(w_i + \sum_{j=1}^{S} I(i,j) * Prompt(w_j))\right)_{w_i \in X},$$

where

- $Prompt(x)$ is a learnable linear (affine) transformation which transforms the head embedding.
- $I(i,j) = \begin{cases} 1, & \text{if } w_j \text{ is the predefined head of } w_i \\ 0, & \text{otherwise} \end{cases}$

The function $I(i,j)$ is an input to the model and it behaves as a matrix which indicates the predefined arcs. The final modified encoder layer looks like this:

$$Layer(X) = O_{ffn}(O_{pa}(O_{attn}(X))).$$

The graphical representation of the modified Transformer block is depicted in Fig. 1.

The Multi Head Attention and FFN parameters are initialized from the weights of the pre-trained model, while the prompt attention parameters are randomly initialized and later learned by fine-tuning. The current implementation allows for a prompt attention sublayer only in some pre-selected layers in the BERT architecture. In this way we could use it only for some of BERT layers. Adding the prompt attention to the last few layers of the model produces best results. We prove this by performing experiments with different settings. In our opinion the reason for this is as follows: adding it to more layers of BERT introduces too many newly initialized parameters and they require longer training.

After the modifications, the head classification layer from 3.1 is appended to the model and it is fine-tuned by an end-to-end training.

3.3 Relation Classification — (LAS)

After receiving the structure of the syntax tree, another model is trained to predict the labels of the word - head arcs.
Let $h_i = Model(w_i)$ and $h_j = Model(w_j)$ be the embeddings of the words w_i and w_j and let w_j be the head of w_i.
The number of relation classes are of a fixed size, so a linear (affine) classifier can be used. The embeddings of the word and its head (predicted in the previous step) are concatenated, then passed through the transformation. The distribution across the possible classes for the relation between w_i and its head w_j is $c_i = Classifier(Concat(h_i, h_j))$. The model is end-to-end trained with cross-entropy loss. Currently the addition of some predefined information for the label classification is outside the scope of our work.

4 Experiments

In this section we present the experiment settings that were used to evaluate the new dependency architecture as well as the results from the different experiments. We performed experiments with two Universal Dependency Treebanks: *BTB Bulgarian Treebank* (Osenova and Simov 2015) and the GUM English Treebank (Zeldes, 2017). The Bulgarian treebank was selected because we are mainly interested in Dependency Parsing for Bulgarian. Also, we have access to many language resources and tools for Bulgarian like Chunk grammars for recognition of noun chunks, verbal complex chunks, prepositional chunks, lexicon with MWEs (still quite modest as a coverage). By performing experiments also for English, we wanted to provide some initial evidence that our architecture is not language specific. The GUM English Treebank was selected because its size is similar to that of BTB Bulgarian Treebank.

For the experiments with the Bulgarian parser we used a pre-trained BERT model with 355M parameters as an encoder which produced the initial embeddings of the tokens. The BERT model was trained by us on 20B of Bulgarian tokens. Our pre-training dataset consists of mainly Web data, literature, administrative and scientific documents, as well as Wikipedia articles. The model was trained for 3 epochs and the pre-training took 23 hours for a single epoch on 16 Nvidia A100s. The models will be uploaded on Huggingface.

For the experiments with the English parser, BERT large uncased was used (Devlin et al., 2018) because the model architecture is similar to our pre-trained BERT. The difference in the parameter count comes from the bigger embedding layer because of the larger vocabulary size of our model.

The notion of $T_P \subseteq T$ — the set of "sure" arcs in the dependency tree T, can easily extended to a whole treebank by applying the same procedure to each sentence in a given treebank. We denote the set of arcs for the whole treebank as $T_P(TreeBankName)$ with additional superscripts if necessary. For Bulgarian we adopted the available constituent-based cascaded chunk grammars where each rule was applied over each sentence annotated with grammatical features (the XPOS column of the CoNLLU format was used as defined for all Universal Dependency treebanks). The rules are ordered and applied according to the specified order on the basis of the result from the previous rules. A very simple example is the following one: if the current sentence contains a preposition (R) and a noun chunk ($NChunk$), then the following rule can be applied:

$R, NChunk \rightarrow PChunk$

When the whole grammar is applied, the arcs within each of the chunks are selected for the corresponding sentence. For example, for the above PChunk we could predict an arc from the preposition to the head of the $NChunk$ with label "`case`". The arcs and their labels could be defined uniquely on the basis of the chunks and grammatical features of the words in them.

The lexicon of the MWEs contains a uniform rep-

Model	Training set	$T_P(BTB)^{ChMWE}$		$T_P(BTB)^0$		$T_P(BTB)^{20}$	
		UAS	**LAS**	**UAS**	**LAS**	**UAS**	**LAS**
Corrected Argmax	$T_P(BTB)^0$	0.9640	0.9361	0.9614	0.9335	0.9694	0.9409
Corrected MST	$T_P(BTB)^0$	0.9640	0.9361	***0.9615***	***0.9337***	0.9695	0.9410
Prompted-10	$T_P(BTB)^{10}$	0.9655	0.9370	0.9626	0.9340	0.9690	0.9400
Prompted-20	$T_P(BTB)^{20}$	0.9641	0.9360	0.9606	0.9324	0.9700	0.9411
Prompted-0-40	$T_P(BTB)^{0+40}$	**0.9672**	**0.9392**	0.9640	0.9362	0.9718	0.9433
Prompted-ChMWE	$T_P(BTB)^{ChMWE}$	0.9655	0.9374	0.9510	0.9231	0.8307	0.8665

Table 1: Accuracy of UAS and LAS of the models on the UD_Bulgarian-BTB test set with different subsets of predefined arcs. The different models are trained on different training sets. The first two models were trained on the treebank without any prompting arcs. For these two models the MST ones are performing generally better. Thus, we selected *Corrected MST* as a baseline model (highlighted in bold and italics) because it achieved the best result on the treebank without any prompting arcs. As the best new model we selected *Prompted-0-40* because it achieved the best result (highlighted in bold) over $T_P(BTB)^{ChMWE}$ — ($ChMWE$ = **Ch**unk grammars and **MWE** lexicon). . This is a realistic scenario, because the prompting arcs are produced by shallow grammars and the lexicons which could be applied over new texts. This model also produced better results over the treebank without prompting arcs. *Prompted-0-40* produced even better results (underlined in the table) over the test set with 20 % random prompting arcs. But this is an unrealistic scenario because we do not have reliable sources for such prompting arcs.

resentation of each MWE which contains not only the strings, but also dependency relations for the structure of the MWE and some grammatical features of its internal elements — see (Osenova and Simov, 2024). Here only MWEs that are realized continuously in the text, and which are unambiguous are used.

The set of arcs selected in this way for the universal BTB treebank is $T_P(BTB)^{ChMWE}$ It contains around 25 % of all arcs in the treebank. When there are not available grammars, lexicons, or just for experiments appropriate sets of arcs could be selected randomly from the treebank itself. In these cases we could have sets such as: $T_P(BTB)^{10}$, $T_P(BTB)^{20}$, $T_P(BTB)^{30}$ containing 10 %, 20 %, 30 % and so on of the arcs in the treebank. Similarly, $T_P(GUM)^{10}$, $T_P(GUM)^{20}$, $T_P(GUM)^{30}$ for the English treebank. In some cases we also use $T_P(BTB)^0$ for the treebank without any prompting arcs selected. Also we use $T_P(BTB)^{0+40}$ to denote the shuffled union of two copies of the treebank: one without any prompting arcs and one with 40 % of prompting arcs.

These sets of arcs could be used during the training, validation and testing of the corresponding parsers. Obviously, the analyses of the new texts will require shallow grammars and/or lexicons of MWEs. If not available, the parser will be used without the predetermined sure arcs.

When the predefined sets of arcs are used to influence only the final decisions of the head selection algorithms (as opposed to injecting data into the model) we call the setup *Corrected models*. Thus, we have *Corrected Argmax* and *Corrected MST models*. As it was mentioned before, for a baseline of the models we use *Corrected MST models*. The corrected Argmax will be reported in order to demonstrate the performance of the parser model strictly by itself.

Also, the fact that in most of cases *Corrected Argmax* and *Corrected MST* produced very close results is an evidence that the transformer model does some reasoning which ensures a tree-like structure of the output graph. Our intuition is that this reasoning happens in the last layers of the encoder, just before the head selection layer, since the head selection layer predicts the head of each word independently. Thus no information sharing can happen there. This motivates the decision to incorporate outside information about the tree in these layers.

4.1 Training and testing set-up

We considered 3 training and testing set-ups:

- Training and testing with predefined arcs produced by arbitrary fixed size subsets of all arcs in the treebanks - ($T_P(BTB)^{10}$, $T_P(BTB)^{20}$ and so on).
- Training and testing with predefined arcs produced by a shallow grammar parser and a lexicon ($T_P(BTB)^{ChMWE}$) — only for Bulgarian.
- Training with predefined arcs produced by ar-

bitrary subsets ($T_P(BTB)^k$) and testing with predefined arcs produced by a shallow grammar parser and a lexicon ($T_P(BTB)^{ChMWE}$) — again only for Bulgarian.

Using an arbitrary predefined subset of arcs as prompts is sufficient to train the model to recognize the prompts and produce good results when testing with both — arbitrary or custom predefined arcs. The ability to train with an arbitrary subset of prompts is important in case of insufficient linguistic resources.

The fine-tuning training is done for 10 epochs with a learning rate of 5e-5 with linear decay and batch size of 384. The fine-tuning takes around 5 minutes on 8 Nvidia A100s. The best performing model checkpoint over the 10 epochs on the validation set is selected.

4.2 Results

In this subsection we present some results from the experiments. In Table 1 the evaluation of Bulgarian parsers is given. The best performing Prompted model for Bulgarian on UAS was trained on a shuffled union of the BTB train set with no predefined arcs and the BTB train set with 40% predefined arcs ($T_P(BTB)^{0+40}$), which enables the model to 'see' sentences without any predefined arcs during training. A paired t-test over 10 training sessions with different random seeds for weight initialization was done and it shows that the improvement in accuracy of the *Prompted models* in comparison to the *Corrected MST* is statistically significant and thus not a product of lucky weights initialization.

Different experiments were made regarding the size of the set of predefined arcs during training and the number of modified encoder layers: We found that the size of the predefined subset should be neither very large nor very small: if too small (under 5% of all arcs) — the model cannot learn the meaning of the prompts and does not use the predefined information. If too large (more than 50%) — the model cannot learn to do parsing on its own. We found that 20% of predefined arcs is an optimal overall size. In addition to the size, it is important to be mentioned that when the prompts are selected randomly, they also belong to different kinds of arcs representing different phenomena within the dependency trees. Having different types of prompt arcs enables the model to better generalize over the meaning of the prompts. This is one explanation why the set of arcs, produced by the shallow grammars and the lexicons ($T_P(BTB)^{ChMWE}$) is not so good for fine-tuning of the dependency graphs comparing to the set with 20 % randomly selected arcs ($T_P(BTB)^{20}$). This is a consequence of the nature of the shallow grammars and most of the MWEs in our lexicon.

The number of the modified layers also matters. We made experiments with adding the prompt attention to half or even all layers, but doing so introduces too many new parameters thus making the training require more examples. In our case the models with modified 2 to 4 of the last layers performed best. The results reported in this paper were produced by models with attention modification considered only for the last 4 out of the 24 layers of BERT (layers 21-24).

The quality of the pre-trained encoder model is also important. Our previous experiments for Bulgarian were done with a BERT model pre-trained on a significantly smaller dataset (4B tokens). While the improvement by adding the modification was still present, the general scores were lower.

In Table 2 the evaluation of the English parsers is given. The results show that not only the improvement with the proposed modification is maintained but there is even some improvement in the case when there are no predefined arcs.

5 Manual Evaluation and Discussion

We performed some manual comparison of the result from the baseline model *Corrected MST* and the best model **Prompted-0-40** for Bulgarian. The two models are correct with respect to the fixed expressions. Thus, the influence of adding some preliminary syntactic knowledge seems to affect the overall analyses and help in cases of correct head identification and direction as well as other phenomena like the PP attachment, apposition relations, etc.

Here two cases are considered: a) the baseline makes errors while the best model is correct, and b) in the opposite direction - the best model makes errors while the baseline is correct.

The baseline makes errors. When inspecting the errors of the baseline where the best model has taken correct decisions, the following main cases have been identified:

- *wrong head direction*: for example, the subject of a copula should be dependant on the

Model	Training set	$T_P(GUM)^{20}$		$T_P(GUM)^0$	
		UAS	**LAS**	**UAS**	**LAS**
Corrected Argmax	$T_P(GUM^0)$	0.9436	0.9246	0.9299	0.9125
Corrected MST	$T_P(GUM^0)$	0.9447	0.9256	***0.9308***	***0.9133***
Prompted-10	$T_P(GUM)^{10}$	0.9467	0.9273	0.9321	0.9143
Prompted-20	$T_P(GUM)^{20}$	**0.9471**	**0.9280**	0.9310	0.9138

Table 2: The accuracy of UAS and LAS of the models on the UD_English-GUM test set with different subsets of pre-defined arcs. The experiments reported here only demonstrate that the behaviour of the models follows the same pattern. The models trained on treebanks augmented with prompting arcs achieve better results even on treebank data without prompting arcs.

content word of the copula predicative but it erroneously was analyzed as depending on the copula

- *wrong head selection*: for example, in Bulgarian NN construction with the first noun indicating quantity, the head is the first noun, while in the baseline the second one was chosen.
- *wrong head assignment*: for example, the subject should be related to the main verb of a sentence but it was assigned to the modal verb instead; in an appositive structure the modifier of the head noun in the dependant structure is wrongly attached to the head of this dependant structure
- *wrong root assignment*: for example, in complex sentences, the baseline assigns the root relation to both verbs — in the main sentence as well as in the clause
- *wrong PP attachment*: for example, instead of depending on the noun, the head of the PP is made dependant on the verb.

The best model makes errors. When inspecting the errors of the best model where the baseline had taken correct decisions, the following main cases have been identified:

- *wrong head direction*: the same error as in the baseline error list
- *wrong head assignment*: for example, in more embedded clauses, the last verb is wrongly attached to the very initial one instead of the nearest governor
- *wrong PP attachment*: the same error as in the baseline error list
- *wrong non-PP attachment*: for example, the adverb is adjacent to the preceding noun but has to be attached to the following verb. However, it was wrongly attached to the noun; when the complementizer 'che' (that) is used in non-typical for it structures like after the negative particle 'ne' (not), the complementizer is wrongly attached to the negative particle instead of to the verb

As it can be seen from the observations above, both models generally make identical types of errors. At the same time it seems that the best one has more attachment-related issues while the baseline — more head-related ones. From the statistics over the errors it can be seen that the baseline makes more errors per category than the best one. Both models have almost the same difficulties with the following labels: *obl* and *advmod*. Thus, all adverbials — despite being expressed by adverbs or nominals, cause problems towards the proper analyses. Also, it seems that the processing of the relation *obj* is easy for the best model while not for the baseline; the processing of the relation *discourse* is easy for the baseline while not for the best model.

The manual validation of the results from the two models shows that the extension of the transformer architecture with the new prompt attention layer improves the general performance of the head selection model. But it also shows that the improvement is not an extension of the baseline model. Instead, the extended model covers a different part of the search space. Thus we plan to address this discrepancy in several ways: (1) through the improvement of the prompt attention layer by including more linguistic information such as higher order arc information, grammatical features, shallow semantic information — ontological information for NEs, terms and key words, where this information is reliable; (2) through the extension of the treebank with

new sentences selected using some active learning procedure. (3) through the improvement of the shallow grammar and the coverage of the MWE lexicon as well as the related algorithms for their better prediction and consequent recognition in text.

6 Conclusion and Future Work

In this work we extended the standard transformer block architecture with a new *prompt attention* layer which incorporates the information from some external knowledge sources like shallow grammars and MWE lexicons. In this way the BERT-based dependency parsing model was internally modified to produce a better dependency parsing model. Here some experimental settings were described where the inclusion of shallow syntactic knowledge and knowledge from MWE lexicons improves the parsing model for Bulgarian. Our assumption is that this architecture would be applicable to any other language. To initially prove this, we also performed experiments with the English UD treebank — GUM.

In our future work we plan to use deeper syntactic knowledge as well as improved shallow syntactic knowledge and semantic information — not only during the fine-tuning stage but also during the pre-training. We plan to make experiments with some variant of the multilingual dependency parsing where the models are simultaneously trained on more than one UD treebank.

Acknowledgments

The reported work has been supported by CLaDA-BG, *the Bulgarian National Interdisciplinary Research e-Infrastructure for Resources and Technologies in favor of the Bulgarian Language and Cultural Heritage, part of the EU infrastructures CLARIN and DARIAH*. We also acknowledge the provided access to *the e-infrastructure of the Centre for Advanced Computing and Data Processing* (the Grant No BG05M2OP001-1.001-0003).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. *Preprint*, arXiv:1611.01734.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's Graph-based Neural Dependency Parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Voula Giouli and Verginica Barbu Mititelu, editors. 2024. *Multiword expressions in lexical resources*. Number 6 in Phraseology and Multiword Expressions. Language Science Press, Berlin.

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

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, page 216–220, USA. ACL.

Petya Osenova and Kiril Simov. 2015. Universalizing BulTreeBank: a linguistic tale about glocalization. In *The 5th Workshop on Balto-Slavic Natural Language Processing*, pages 81–89, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Petya Osenova and Kiril Simov. 2024. Representation of multiword expressions in the Bulgarian integrated lexicon for language technology. In Voula Giouli and Verginica Barbu Mititelu, editors, *Multiword expressions in lexical resources: Linguistic, lexicographic, and computational perspectives*, Phraseology and Multiword Expressions, chapter 6, pages 117—-146. Language Science Press, Berlin.

Saziye Betül Özates, Arzucan Özgür, Tunga Güngör, and Balkiz Öztürk. 2020. A hybrid approach to dependency parsing: Combining rules and morphology with deep learning. *CoRR*, abs/2002.10116.

Yikang Shen, Shawn Tan, Alessandro Sordoni, Siva Reddy, and Aaron Courville. 2021. Explicitly modeling syntax in language models with incremental parsing and a dynamic oracle. In *Proc. of the 2021 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 1660–1672, Online. ACL.

Ryo Yoshida, Taiga Someya, and Yohei Oseki. 2024. Tree-planted transformers: Unidirectional transformer language models with implicit syntactic supervision. *Preprint*, arXiv:2402.12691.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Houquan Zhou, Yang Hou, Zhenghua Li, Xuebin Wang, Zhefeng Wang, Xinyu Duan, and Min Zhang. 2023. How well do large language models understand syntax? an evaluation by asking natural language questions. *Preprint*, arXiv:2311.08287.