Parsing as Tagging

Robert Vacareanu[1], George C. G. Barbosa[2], Marco A. Valenzuela-Escárcega[2], Mihai Surdeanu[2]
[1] Technical University of Cluj-Napoca, [2] University of Arizona
[1]Cluj-Napoca, Cluj, Romania, [2]Tucson, AZ, USA
robert.vacareanu@gmail.com
{gcgbarbosa, marcov, msurdeanu}@email.arizona.edu

Abstract

We propose a simple yet accurate method for dependency parsing that treats parsing as tagging (PaT). That is, our approach addresses the parsing of dependency trees with a sequence model implemented with a bidirectional LSTM over BERT embeddings, where the "tag" to be predicted at each token position is the relative position of the corresponding head. For example, for the sentence *John eats cake*, the tag to be predicted for the token *cake* is `-1` because its head (*eats*) occurs one token to the left. Despite its simplicity, our approach performs well. For example, our approach outperforms the state-of-the-art method of (Fernández-González and Gómez-Rodríguez, 2019) on Universal Dependencies (UD) by 1.76% unlabeled attachment score (UAS) for English, 1.98% UAS for French, and 1.16% UAS for German. On average, on 15 UD languages, our method with minimal tuning performs comparably with this state-of-the-art approach, being only 0.16% UAS, and 0.82% LAS behind.

Keywords: dependency parsing, sequence methods

1. Introduction

There exists a trend in syntactic dependency parsing towards simpler and simpler approaches. For example, early dependency parsing algorithms had a runtime complexity of $O(n^3)$ (Eisner, 1996). These were followed by maximum spanning tree algorithms with a complexity of $O(n^2)$ (McDonald et al., 2005), and shift-reduce methods, which are linear in sentence length ($O(n)$) (Nivre, 2003). This quest for simplicity continued with the recent works of Ma et al. (2018) and Fernández-González and Gómez-Rodríguez (2019), which replaced the shift-reduce algorithm with pointer networks, which are also linear but are arguably simpler than the shift-reduce algorithm.

This evolutionary trend begs the question: can we simplify dependency parsing further, without losing considerable performance? This paper indicates that the answer is yes. We propose an extremely simple method for dependency parsing that treats parsing as tagging (PaT). That is, our approach addresses the parsing of dependency trees with a sequence model, where the "tag" to be predicted at each token position is the relative position of the corresponding head. For example, for the sentence *John eats cake*, the tag to be predicted for the token *cake* is `-1` because its head (*eats*) occurs one token to the left.[1]

The contributions of this work are:

(1) We propose a simple approach for dependency parsing that operates in three stages, which are all widely used in other natural language processing (NLP) approaches. First, we encode the input tokens using a combination of randomly initialized word embeddings, contextualized embeddings (Devlin et al., 2018), character-level embeddings generated using a convolutional neural network (CNN), and part-of-speech (POS) embeddings. Second, these representations are fed into a BiLSTM. Finally, using the BiLSTM's hidden states, we predict the relative position of the head for each token in the sentence and the label of the corresponding dependency.

(2) Despite its simplicity, we show that our approach performs well. For example, our approach outperforms the state-of-the-art method of Fernández-González and Gómez-Rodríguez (2019) on Universal Dependencies (UD) by 1.76% unlabeled attachment score (UAS) and 1.26% labeled attachment score (LAS) for English, 1.98% UAS and 1.65% LAS for French, and 1.16% UAS and 0.45% LAS for German. On average, on 15 UD languages, our method performs near this state-of-the-art approach, being only 0.16% UAS and 0.82% LAS below.

(3) An ablation analysis indicates that the BERT contextualized embeddings and the word-level BiLSTM have considerable contributions to performance, confirming earlier work that indicated the importance of contextualized embeddings for many NLP tasks (Devlin et al., 2018), and that LSTMs capture grammatical structure (Linzen et al., 2016; Kuncoro et al., 2018). As removing either of these components impacts performance negatively, this suggests that we are approaching the limits of simplicity for this task.

2. Related Work

Early algorithms for automatic dependency parsing such as the dynamic programming approach proposed by Eisner (1996) had a complexity of $O(n^3)$. Currently, the two primary approaches, i.e., the graph-based maximum spanning tree approach proposed by McDonald et al. (2005) and the transition based approach formalized by Nivre (2003), have complexities of $O(n^2)$ and $O(n)$, respectively. Many variants of these lower-complexity transition-based approaches have been proposed. Yamada and Matsumoto (2003) trained a support vector machine to direct a shift-reduce parser. Chen and Manning (2014) encode features as embeddings and feed them to a multi-layer perceptron.

[1]Using the same approach, we also predict the label of the corresponding dependency, e.g., `dobj` in this example.

Kiperwasser and Goldberg (2016) use a BiLSTM to learn feature representations, and use these to encode the parser state. Xipeng (2009) proposed a simpler method, where the dependency parsing task is transformed into a sequence labeling problem using conditional random fields. More recently, Ma et al. (2018) proposed a stack-pointer network, which uses information from the sentence as a whole. Fernández-González and Gómez-Rodríguez (2019) introduced a left-to-right parsing approach with a pointer network, reducing the number of transitions required by Ma et al. (2018) from $2n - 1$ to n. Our approach continues this simplification trend with a method that reduces parsing to a simple sequence modeling task, similar with the approaches of (Strzyz et al., 2019) and (Li et al., 2018), albeit with a much simpler architecture and better overall performance. Li et al. (2018) propose an encoder-decoder architecture with an attention layer; Strzyz et al. (2019) propose an encoder-decoder architecture with a more complex encoding scheme. We propose a much simpler architecture, consisting of only a BiLSTM operating on top of contextualized embeddings. Similar to our direction, Hewitt and Manning (2019) showed that syntax trees are embedded in a linear transformation of the BERT and ELMo embeddings. However, they did not consider labels and the evaluation was done only on undirected unlabeled attachment score (UUAS) after generating a minimum spanning tree on the predicted distance graph.

In the pool of exciting methods that simplify the parsing task, our approach is closer in spirit to the approach of Zhang et al. (2017) for dependency parsing, and Marcheggiani et al. (2017) for semantic role labeling. Both these works also encode the tokens in a sentence using an LSTM. However, they operate over *pairs* of words to predict a syntactic dependency between a modifier and head (Zhang et al., 2017), or a token's semantic role given a predicate (Marcheggiani et al., 2017). Further, (Marcheggiani et al., 2017) separately encode the sentence with an LSTM for each predicate. Our approach is simpler and faster, i.e., we encode the sentence just once with the LSTM, and we predict the relative position of head words rather than relying on all possible pairs of words.

3. PaT: Parsing as Tagging

Our approach casts the task of dependency parsing to one of sequence tagging. Since in the dependency tree formalism, each token has a *single* head, which may be another token in the sentence or `ROOT`, we can naturally map the parsing task into a tagging problem. That is, the "tag" of each token becomes the relative position of its head ($relpos$), computed as follows:

$$relpos = \begin{cases} 0, & \text{if head is ROOT} \\ head - id, & \text{otherwise} \end{cases} \tag{1}$$

where id is the absolute position of the current token (starting at 1) (Buchholz and Marsi, 2006), and $head$ is the absolute position of the corresponding head word.

As an example, consider the sentence in Figure 1. The relative position of the head of *cake* is -1 because its head (*eats*) is one word to the left. A relative position of 0 indicates that the token is headed by the `ROOT`. To predict the dependency parse, our architecture straightforwardly encodes the sentence, and for each token predicts these relative positions, as well as the label of the directed edge between them.

Because we framed this task as tagging, we need to constrain prediction to a finite number of possible relative positions. We empirically chose a range of $(-50, 50)$, which accounts for 99.9% of the English dependencies in the Universal Dependencies training dataset. We use the same range for all languages.

3.1. Token Representations

Given an input sentence s, consisting of n tokens $t_1, \ldots, t_n$, we represent each token t_i by a vector e_i, which is the concatenation of (a) the pretrained BERT representation[2] (Devlin et al., 2018) of the token; (b) the word embedding (*we*) for the token; (c) the character CNN encoding (*ce*) of the token; and (d) its part-of-speech (POS) embedding (*pos*):

$$e_i = \left[t_i^{(bert)} \cdot t_i^{(we)} \cdot t_i^{(ce)} \cdot t_i^{(pos)} \right] \tag{2}$$

For the character-level encoding, we use a character CNN with max pooling, which has been shown to be useful by Lample et al. (2016). We learned the embeddings for part-of-speech tags and words, which were initialized using Xavier initialization (Glorot and Bengio, 2010).

3.2. Architecture

Our proposed architecture, as shown in Figure 1, consists of a BiLSTM (Hochreiter and Schmidhuber, 1997) that operates over our token representations e_i, producing the corresponding hidden state h_i. For each token t_i we use h_i in two ways: to predict the relative position of the head, and to predict the label of the corresponding dependency relation. For the relative position, we pass h_i into a multi-layer perceptron (MLP), followed by a linear layer and a softmax to get a distribution over the possible relative positions. For the corresponding dependency label, we concatenate the MLP outputs corresponding to t_i and its predicted head t_i^{head} and pass them to another linear layer and softmax to produce a distribution over the dependency labels.

The objective function of the model is calculated by the cross entropy (Nasr et al., 2002) between predicted and observed values of both the dependency labels and relative positions.

3.3. Cycle Detection

Note that there is nothing in the above architecture that prevents our approach from generating cycles. To control for this, we explore three post-processing options for handling cycles:

(1) No cycle removal, leaving the output of the model unchanged.

(2) Greedy cycle removal, where we globally sort the predicted dependencies based on weight. Then, we incrementally add to the output tree the dependency with the highest

[2] We used pre-trained "BERT-Base Uncased" model for english and "BERT-Base Multilingual Cased (new)" model for other languages. We do not fine-tune.

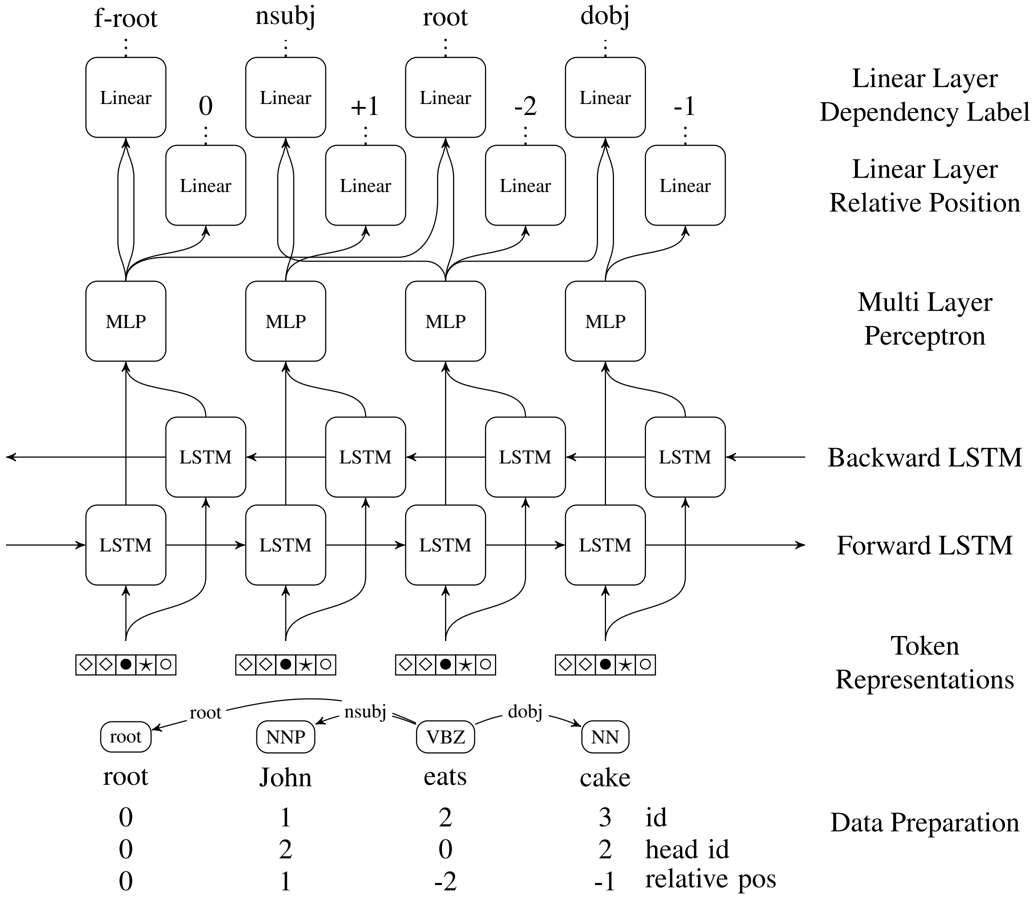

Figure 1: PaT architecture: we feed a concatenation of several representations (Section 3.1.) to a BiLSTM and then a multilayer perceptron (MLP). The MLP output is passed through a linear layer to predict the relative positions. Once a head has been selected, we concatenate the MLP output for the dependent and the predicted head, and pass it to another linear layer to predict the corresponding dependency label.

probability which does not add a cycle, until all tokens are covered.

(3) Optimal cycle removal, which finds a maximum spanning tree (MST) using the Chu-Liu-Edmonds algorithm (Edmonds, 1967).

4. Experiments and Results

We used the same datasets as Fernández-González and Gómez-Rodríguez (2019) and Ma et al. (2018). We tested our model on the Stanford Dependencies (de Marneffe and Manning, 2008) (SD) conversion of the Penn Treebank (Marcus et al., 1993) and on 15 languages from the Universal Dependencies (UD) Treebank. For SD we used the standard splits and the predicted part-of-speech tags. For UD we used the same set up as Fernández-González and Gómez-Rodríguez (2019),[3] with the addition of Arabic, Estonian, and Japanese.

Table 1 lists the unlabeled (UAS) and labeled (LAS) accuracies of our model (averaged over 3 runs) for 15 languages from the UD Treebank, compared against the state-of-the-art method of Fernández-González and Gómez-Rodríguez (2019). The hyper parameters were tuned on English UD, by exploring 100 configurations. The same hyper parameters were used for *all* languages in this table.

In Table 2 we show the performance of our method after tuning the hyper parameters for *each* language. For each language, we selected the best performing model from a set of 15 configurations. The same search space was used for all languages.

In our experiments, we used early stopping to halt training if there is no improvement for 3 epochs. In some instances, the training stops too early, resulting in a bigger standard deviation in some cases (e.g., Italian), or sub-optimal performance in others. For future experiments, we recommend either to use a warm-up phase of 10–15 epochs, or to increase the early stopping threshold to 5.

Despite the simple architecture, the two tables show that PaT outperforms the method of Fernández-González et al. on six languages for UAS, and on four for LAS; on average, on the 15 languages, its performance is 0.16% UAS and 0.82% LAS behind this state-of-the-art parser.

Even though PaT does not enforce the production of acyclic structures, relatively few parse trees produced by PaT contain cycles: between 0.47% for Japanese and 17.20% for Arabic, for an average of 5.11% $\pm$ 1.11. Further, we observe that the simple, greedy cycle removal performs similarly to the optimal MST algorithm, which is probably due

[3] In particular, we also used version 2.2 of UD, to facilitate comparison with previous work.

Language	Fernández-González et al.		This work						
			% of sentences with cycles	Without cycle removal		With greedy cycle removal		With optimal cycle removal	
	UAS	LAS		UAS	LAS	UAS	LAS	UAS	LAS
ar	**87.93** ± 0.02	**83.47** ± 0.03	17.64 ± 2.47	84.45 ± 0.21	78.94 ± 0.3	84.38 ± 0.26	78.84 ± 0.32	84.36 ± 0.28	78.75 ± 0.36
bu	**94.42** ± 0.02	**90.70** ± 0.04	3.05 ± 1.42	94.23 ± 0.17	89.77 ± 0.27	94.28 ± 0.14	89.74 ± 0.26	94.29 ± 0.17	89.71 ± 0.27
ca	**94.07** ± 0.06	**92.26** ± 0.05	6.41 ± 0.50	93.80 ± 0.03	91.43 ± 0.31	93.80 ± 0.05	91.40 ± 0.32	93.80 ± 0.05	91.36 ± 0.34
cs	**94.19** ± 0.04	**91.45** ± 0.05	2.85 ± 0.87	93.77 ± 0.09	90.14 ± 0.15	93.77 ± 0.09	90.10 ± 0.16	93.78 ± 0.08	90.07 ± 0.16
de	87.28 ± 0.07	82.99 ± 0.07	5.73 ± 1.68	88.37 ± 0.30	**83.44** ± 0.44	**88.44** ± 0.26	83.41 ± 0.41	88.43 ± 0.29	83.35 ± 0.42
en	90.93 ± 0.11	88.99 ± 0.11	2.88 ± 1.16	92.64 ± 0.09	**90.25** ± 0.25	**92.69** ± 0.05	90.21 ± 0.26	92.68 ± 0.06	90.15 ± 0.26
es	93.23 ± 0.03	**91.28** ± 0.02	7.61 ± 1.09	93.33 ± 0.15	90.89 ± 0.19	**93.35** ± 0.13	90.85 ± 0.18	93.35 ± 0.13	90.83 ± 0.17
et	**87.72** ± 0.07	**84.98** ± 0.12	5.02 ± 0.50	87.12 ± 0.14	83.48 ± 0.3	87.17 ± 0.14	83.45 ± 0.31	87.16 ± 0.14	83.39 ± 0.32
fr	90.97 ± 0.09	88.22 ± 0.12	5.20 ± 0.49	92.92 ± 0.35	**89.87** ± 0.24	92.91 ± 0.34	89.84 ± 0.25	**92.95** ± 0.29	89.80 ± 0.25
it	94.28 ± 0.06	**92.48** ± 0.02	3.80 ± 2.28	94.25 ± 0.81	91.93 ± 0.99	**94.29** ± 0.76	91.93 ± 0.98	94.26 ± 0.78	91.90 ± 1.01
ja	94.03 ± 0.05	91.54 ± 0.04	0.47 ± 0.1	94.25 ± 0.27	92.09 ± 0.35	**94.26** ± 0.28	**92.09** ± 0.36	94.26 ± 0.28	92.09 ± 0.36
nl	**93.23** ± 0.09	**90.74** ± 0.08	3.66 ± 0.35	92.87 ± 0.13	89.51 ± 0.20	92.93 ± 0.16	89.49 ± 0.20	92.92 ± 0.18	89.46 ± 0.20
no	**95.23** ± 0.06	**93.99** ± 0.07	1.81 ± 0.42	94.65 ± 0.12	92.88 ± 0.19	94.66 ± 0.13	92.86 ± 0.20	94.66 ± 0.13	92.84 ± 0.19
ro	**91.58** ± 0.08	**86.00** ± 0.07	7.22 ± 2.59	90.24 ± 0.23	83.45 ± 0.36	90.28 ± 0.18	83.40 ± 0.32	90.25 ± 0.25	83.34 ± 0.37
ru	**94.71** ± 0.07	**93.38** ± 0.09	3.27 ± 0.70	93.91 ± 0.06	91.98 ± 0.12	93.93 ± 0.07	91.95 ± 0.12	93.92 ± 0.07	91.91 ± 0.12

Table 1: Accuracy comparison between PaT and the best performing models of Fernández-González et al. on the test partitions of 15 languages from the Universal Dependencies Treebank. The bold font indicates the best performing model on each language. PaT's reported results are the average and the standard deviation (stdev) over three runs. The *same parameters, tuned for English*, were used for all languages.

Language	Fernández-González et al.		This work						
			% of sentences with cycles	Without cycle removal		With greedy cycle removal		With optimal cycle removal	
	UAS	LAS		UAS	LAS	UAS	LAS	UAS	LAS
ar	**87.93** ± 0.02	**83.47** ± 0.03	17.20 ± 2.55	84.75 ± 0.21	78.96 ± 0.28	84.74 ± 0.24	78.9 ± 0.32	84.72 ± 0.25	78.85 ± 0.34
bu	**94.42** ± 0.02	**90.70** ± 0.04	2.21 ± 0.33	94.04 ± 0.33	89.56 ± 0.21	94.07 ± 0.30	89.56 ± 0.20	94.06 ± 0.30	89.50 ± 0.23
ca	**94.07** ± 0.06	**92.26** ± 0.05	5.77 ± 0.54	93.74 ± 0.17	91.56 ± 0.20	93.77 ± 0.20	91.55 ± 0.20	93.76 ± 0.20	91.51 ± 0.21
cs	**94.19** ± 0.04	**91.45** ± 0.05	2.85 ± 0.87	93.77 ± 0.09	90.14 ± 0.15	93.77 ± 0.09	90.10 ± 0.16	93.78 ± 0.08	90.07 ± 0.16
de	87.28 ± 0.07	82.99 ± 0.07	4.29 ± 0.89	88.70 ± 0.05	**83.80** ± 0.15	88.78 ± 0.07	83.80 ± 0.15	**88.78** ± 0.06	83.75 ± 0.15
en	90.93 ± 0.11	88.99 ± 0.11	2.88 ± 1.16	92.64 ± 0.09	**90.25** ± 0.25	**92.69** ± 0.05	90.21 ± 0.26	92.68 ± 0.06	90.15 ± 0.26
es	93.23 ± 0.03	**91.28** ± 0.02	7.84 ± 1.48	93.22 ± 0.14	90.80 ± 0.16	**93.26** ± 0.12	90.78 ± 0.16	93.24 ± 0.13	90.75 ± 0.17
et	**87.72** ± 0.07	**84.98** ± 0.12	5.02 ± 0.50	87.12 ± 0.14	83.48 ± 0.3	87.17 ± 0.14	83.45 ± 0.31	87.16 ± 0.14	83.39 ± 0.32
fr	90.97 ± 0.09	88.22 ± 0.12	5.20 ± 0.49	92.92 ± 0.35	**89.87** ± 0.24	92.91 ± 0.34	89.84 ± 0.25	**92.95** ± 0.29	89.80 ± 0.25
it	94.28 ± 0.06	**92.48** ± 0.02	3.17 ± 1.18	94.52 ± 0.52	92.31 ± 0.64	**94.52** ± 0.53	92.31 ± 0.63	94.50 ± 0.50	92.27 ± 0.62
ja	94.03 ± 0.05	91.54 ± 0.04	0.48 ± 0.83	94.35 ± 0.1	92.35 ± 0.06	94.35 ± 0.11	92.35 ± 0.07	**94.36** ± 0.08	**92.35** ± 0.06
nl	**93.23** ± 0.09	**90.74** ± 0.08	4.46 ± 1.02	92.81 ± 0.15	89.51 ± 0.21	92.88 ± 0.14	89.49 ± 0.18	92.87 ± 0.15	89.46 ± 0.18
no	**95.23** ± 0.06	**93.99** ± 0.07	1.87 ± 0.52	94.65 ± 0.05	92.95 ± 0.08	94.67 ± 0.06	92.93 ± 0.10	94.66 ± 0.05	92.91 ± 0.09
ro	**91.58** ± 0.08	**86.00** ± 0.07	5.94 ± 0.52	90.64 ± 0.07	83.71 ± 0.07	90.71 ± 0.05	83.70 ± 0.07	90.72 ± 0.05	83.67 ± 0.07
ru	**94.71** ± 0.07	**93.38** ± 0.09	2.32 ± 0.77	94.12 ± 0.11	92.22 ± 0.15	94.13 ± 0.09	92.21 ± 0.15	94.14 ± 0.08	92.18 ± 0.14

Table 2: Accuracy comparison between tuned PaT and the best performing models of Fernández-González et al. on the test partitions of 15 languages from the Universal Dependencies Treebank. The bold font indicates the best performing model on each language. PaT's reported results are the average and the standard deviation (stdev) over three runs. For each language, we select the *best performing configuration* from a set of 12 possible configurations.

to the above-mentioned fact that only a small percentage of sentences yield structures with cycles.

In Table 3, we list unlabeled and labeled accuracies on the SD test partition, along with the performance of state-of-the-art approaches and the type of method used by each. The reported accuracy for our system is the mean and standard deviation over 5 runs, without cycle removal. As the table shows, PaT outperforms many more complex methods. All in all, on SD, PaT outperforms the best approach by 0.23% LAS, while underperforming the best by less than 0.2% UAS.

Table 4 shows the UAS error rates for English UD based on distance between the dependent and its head. PaT's error rate is small for heads that are within three tokens to their dependents, the most frequent situation, and increases with the distance.

Table 5 shows an ablation analysis performed on the English UD dataset. As shown, eliminating the BiLSTM has the biggest impact on performance, followed by BERT, and POS embeddings. This confirms previous observations that LSTMs capture grammatical structure (Linzen et al., 2016; Kuncoro et al., 2018), and are the critical ingredient to de-

Parser	UAS	LAS	Type
Chen and Manning (2014)	91.80	89.60	Tr
Dyer et al. (2015)	93.10	90.90	Tr
Kiperwasser and Goldberg (2016)	93.10	91.00	Tr
Ballesteros et al. (2016)	93.56	91.42	Tr
Strzyz et al. (2019)	93.67	91.72	Tag
Kiperwasser and Goldberg (2016)	93.90	91.90	Tr
Weiss et al. (2015)	93.99	92.05	Tr
Wang and Chang (2016)	94.08	91.82	G
Cheng et al. (2016)	94.10	91.49	G
Li et al. (2018)	94.11	92.08	Tag
Alberti et al. (2015)	94.23	92.36	Tr
Kuncoro et al. (2018)	94.26	92.06	G
Zhang et al. (2017)	94.30	91.95	G
Qi and Manning (2017)	94.30	92.20	Tr
Fernández-González and Gómez-Rodríguez (2018)	94.50	92.40	Tr
Andor et al. (2016)	94.61	92.79	Tr
Dozat and Manning (2016)	95.74	94.08	G
This work (untuned)	95.85 $\pm$ 0.20	94.60 $\pm$ 0.24	Tag
Ma et al. (2018)	95.87	94.19	Tr
This work (tuned)	95.87 $\pm$ 0.05	**94.66** $\pm$ **0.07**	Tag
Fernández-González and Gómez-Rodríguez (2019)	**96.04**	94.43	Tr

Table 3: PaT performance on Stanford Dependencies, compared to other methods on the test partition of the Penn Treebank. For our approach we used the greedy cycle removal strategy, and we report mean and stdev over 5 runs. Tr, G and Tag indicate transition-, graph-, and tag-based methods, respectively.

pendency parsing.

In Table 6, we discuss the amount of effort in tuning and training each language. As expected, the time is highly dependent on the size of the data, taking from 60 seconds per epoch for Bulgarian to 2100 seconds per epoch for Czech. For English, we selected the best performing model from a set of 100 configurations.

We tuned hyper parameters for the other languages by selecting the best performing model from a set of 12 configurations.

	Distance between head and modifier			
	0–3	4–9	10–19	20+
Error rate (%)	4.86 $\pm$ 0.18	13.91 $\pm$ 0.47	37.85 $\pm$ 3.82	74.19 $\pm$ 6.89
Total deps	18070	3280	516	124

Table 4: PaT error rates (mean and stdev over 3 runs) for the English UD data, grouped by the distance between modifier and head.

	UAS	LAS
Full model	92.69 $\pm$ 0.05	90.25 $\pm$ 0.25
– CNN Char Embeddings	92.04 $\pm$ 0.21	89.76 $\pm$ 0.32
– POS Embeddings	91.42 $\pm$ 0.11	87.96 $\pm$ 0.11
– BERT	88.59 $\pm$ 0.1	86.14 $\pm$ 0.1
– BiLSTM	64.28 $\pm$ 0.38	62.97 $\pm$ 0.4

Table 5: Four different types of ablation tests performed over the English UD dataset. The reported results represent the mean and stdev over 3 runs with greedy cycle removal by removing only one component in each row.

5. Conclusion

We proposed an approach that reframes dependency parsing as a sequence tagging task that relies solely on surface information. Specifically, for each token in a given sentence, we predict the relative position to that token's head, as well as the corresponding dependency label. Our approach achieves state-of-the-art performance on three Universal Dependencies languages and strong performance on nine others. This work suggests that parsing as tagging can serve as a new, simple, yet strong baseline for dependency parsing.

For reproducibility, we release the code behind this work as open source. The software, together with all hyper parameters used, is available at this URL: `https://github.com/clulab/releases/tree/master/lrec2020-pat`.

Language		Configurations	Average epoch time (s)	Data size (MB)
en	(UD)	100	120	11
en	(SD)	12	250	29
ar	(UD)		110	35
bg	(UD)		60	9.1
ca	(UD)		200	24
cs	(UD)		2100	173
de	(UD)		120	16
es	(UD)		360	46
et	(UD)		150	17
fr	(UD)		165	19
it	(UD)		150	15
ja	(UD)		75	7.5
nl	(UD)		150	17
no	(UD)		240	26
ro	(UD)		90	13
ru	(UD)		600	67

Table 6: Amount of effort for tuning PaT. For English, we selected the best performing model from a set of 100 configurations. For every other language, we selected the best performing model from a space of 15 configurations. The same search space was used for all the languages. The training time varies from 60s (Bulgarian) to 2100s (Czech), depending on the data size, on an Nvidia Tesla P4 GPU.

6. Acknowledgments

This work was funded by the Bill and Melinda Gates Foundation HBGDki Initiative. Marco Valenzuela-Escárcega and Mihai Surdeanu declare a financial interest in `lum.ai`. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

7. Bibliographical References

Alberti, C., Weiss, D., Coppola, G., and Petrov, S. (2015). Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1354–1359, Lisbon, Portugal, September. Association for Computational Linguistics.

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August. Association for Computational Linguistics.

Ballesteros, M., Goldberg, Y., Dyer, C., and Smith, N. A. (2016). Training with exploration improves a greedy stack-lstm parser. *CoRR*, abs/1603.03793.

Buchholz, S. and Marsi, E. (2006). Conll-x shared task on multilingual dependency parsing. In *Proceedings of the tenth conference on computational natural language learning*, pages 149–164. Association for Computational Linguistics.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October. Association for Computational Linguistics.

Cheng, H., Fang, H., He, X., Gao, J., and Deng, L. (2016). Bi-directional attention with agreement for dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214, Austin, Texas, November. Association for Computational Linguistics.

de Marneffe, M.-C. and Manning, C. D. (2008). The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dozat, T. and Manning, C. D. (2016). Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Fernández-González, D. and Gómez-Rodríguez, C. (2018). Non-projective dependency parsing with non-local transitions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 693–700, New Orleans, Louisiana, June. Association for Computational Linguistics.

Fernández-González, D. and Gómez-Rodríguez, C. (2019). Left-to-right dependency parsing with pointer networks. *CoRR*, abs/1903.08445.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh et al., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May. PMLR.

Hewitt, J. and Manning, C. D. (2019). A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kuncoro, A., Dyer, C., Hale, J., Yogatama, D., Clark, S., and Blunsom, P. (2018). LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia, July. Association for Computational Linguistics.

Lample, G., Ballesteros, M., Subramanian, S. e., Kawakami, K., and Dyer, C. (2016). Neural Architectures for Named Entity Recognition. *arXiv e-prints*, page arXiv:1603.01360, Mar.

Li, Z., Cai, J., He, S., and Zhao, H. (2018). Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Ma, X., Hu, Z., Liu, J., Peng, N., Neubig, G., and Hovy, E. (2018). Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia, July. Association for Computational Linguistics.

Marcheggiani, D., Frolov, A., and Titov, I. (2017). A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593*.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Nasr, G. E., Badr, E. A., and Joun, C. (2002). Cross entropy error function in neural networks: Forecasting gasoline demand. In *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*, pages 381–384. AAAI Press.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies*, pages 149–160, Nancy, France, April.

Qi, P. and Manning, C. D. (2017). Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–117, Vancouver, Canada, July. Association for Computational Linguistics.

Strzyz, M., Vilares, D., and Gomez-Rodriguez, C. (2019). Viable dependency parsing as sequence labeling.

Wang, W. and Chang, B. (2016). Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, Berlin, Germany, August. Association for Computational Linguistics.

Weiss, D., Alberti, C., Collins, M., and Petrov, S. (2015). Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China, July. Association for Computational Linguistics.

Xipeng, J. F. Q. (2009). A new chinese dependency analysis method based on sequence labeling model. *Computer Applications and Software*, October.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, Nancy, France, April.

Zhang, X., Cheng, J., and Lapata, M. (2017). Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain, April. Association for Computational Linguistics.