# Parsing Universal Dependency Treebanks using Neural Networks and Search-Based Oracle

Milan Straka Jan Hajič Jana Straková Jan Hajič jr.
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{straka,hajic,strakova,hajicj}@ufal.mff.cuni.cz

**Abstract**

We describe a transition-based, non-projective dependency parser which uses a neural network classifier for prediction and requires no feature engineering. We propose a new, search-based oracle, which improves parsing accuracy similarly to a dynamic oracle, but is applicable to any transition system, such as the fully non-projective `swap` system, contrary to dynamic oracles, which are specific for each transition system and usually quite complex. The parser has excellent parsing speed, compact models, and achieves high accuracy without requiring any additional resources such as raw corpora. We tested it on all 37 treebanks of the Universal Dependencies project. The C++ implementation of the parser is being released as an open-source tool.

## 1 Introduction

Transition-based systems were proposed by Yamada and Matsumoto [28] and Nivre [16]. Greedy transition-based parsers are very efficient while achieving reasonably high accuracy, allowing to parse large volumes of data.[1]

An *oracle* is used at training time to map parser configurations to optimal transitions given a gold tree. A classifier is then trained to emulate the oracle predictions.

Initially, transition-based parsers used *static oracles*, which are defined only for configurations from which the complete gold tree can be reached. Recently, Goldberg and Nivre [9, 10], Goldberg et al. [11], Gómez-Rodríguez et al. [13] and Gómez-Rodríguez and Fernández-González [12] improved accuracy of transition-based parsers by utilizing a *dynamic oracle*, which is defined for any parser configuration and predicts transitions leading to a tree most similar to the gold one. Such a dynamic oracle affects only the training speed, not parsing speed. However, a

[1] Beam search can improve parsing accuracy but at a substantially lower speed, cf. e.g. Zhang and Nivre [31].

dynamic oracle is usually more complicated than a static one; for example, the dynamic oracle of Gómez-Rodríguez et al. [13] for a restricted non-projective system has $O(n^8)$ complexity.

In this paper we consider a new *search-based oracle*, which resembles the dynamic oracle in terms of predicting transitions from any parser configuration. However, a search-based oracle utilizes only the classifier being trained, which makes it applicable to any transition system with a static oracle only. Still, parsing accuracy of a search-based oracle is comparable to the dynamic oracle.

Inspired by recent success of distributed word representations in NLP, e.g. in POS tagging (Collobert et al. [3]), machine translation (Devlin et al. [6]), constituency parsing Socher et al. [26] and projective dependency parsing (Chen and Manning [2]), we train a neural network (NN) classifier predicting transitions in a transition-based parser. We utilize the search-based oracle allowing the `swap` operation and thus more accurate fully non-projective parsing. We train our parser on all 37 Universal Dependencies (UD) treebanks version 1.2, showing that high accuracy can be achieved by the new search-based oracle and using a neural network classifier even without additional raw corpora.

The main contributions of this work are:

- a novel search-based oracle which can be used with any transition system, improving the parsing results considerably, comparably to using a dynamic oracle (Sect. 4):
  - notably, the search-based oracle can be applied to the non-projective transition system with the `swap` operation, which enables fully non-projective parsing;
  - the search-based oracle can be used even on top of a dynamic oracle, further improving accuracy;
- a NN-based parser with better accuracy for most of the UD treebanks and substantially improved speed for all of them, while keeping models compact (Sect. 3);
- an open-source C++ parser implementation[2] and parsing models for all 37 treebanks of Universal Dependencies 1.2 [21].

## 2 Transition-based Dependency Parsing

Transition-based dependency parsing computes the dependency tree for a sentence by starting in an initial configuration and performing a sequence of transitions reaching some terminal configuration.

One of the most popular transition systems is the projective stack-based arc-standard system by Nivre [17], which we denote as `stack`. This system employs three types of transitions: *left_arc*$_l$ and *right_arc*$_l$, which add a dependency arc with label $l$, and *shift*, which adds the next input word.

There are also several transition systems that allow parsing of non-projective trees. Attardi [1] introduced transitions to the `stack` system adding dependency

[2] `http://hdl.handle.net/11234/1-1573`

arcs between non-adjacent subtrees. Here we consider a restriction of the original Attardi parser described for example in Gómez-Rodríguez et al. [13], which we denote as `arc2`. The `arc2` system extends the `stack` system by adding transitions $left\_arc\_2_l$ and $right\_arc\_2_l$ which add dependency arcs between non-adjacent nodes. Although only some non-projective trees can be obtained by such transitions, Attardi in [1] notes that the `arc2` system is sufficient to handle almost all cases of non-projectivity in the training data.

The truly non-projective transition system which we call `swap` was proposed by Nivre [18]. It extends the `stack` system by adding the *swap* transition for reordering two nodes. Nivre et al. [20] show that any non-projective tree can be reached while keeping the expected time linear.

# 3 Neural Network Classifier

The architecture of the neural network classifier is similar to that described in Chen and Manning [2].

The input to the network consists of several nodes representing words in the tree being built. Following Zhang and Nivre [31] and Chen and Manning [2], we use a rich set of up to 18 nodes as input: top 3 nodes on the stack, top 3 nodes on the buffer, the first and second leftmost/rightmost children of the top 2 nodes on the stack, and leftmost of leftmost and rightmost of rightmost children of the top 2 nodes on the stack.

Each node is represented using distributed representations of its form, its POS tag and its arc label; the latter only if it has already been assigned.

In the Universal Dependency treebanks, there are three token fields connected to part-of-speech: UPOSTAG (universal part-of-speech tag), XPOSTAG (language-specific part-of-speech tag, which is not present in many treebanks) and FEATS (list of morphological features further refining the universal part-of-speech tag). We use both UPOSTAG and FEATS fields, which improves results considerably, compared to using only UPOSTAG.

The input layer is connected to a hidden layer with *tanh* activation. The output layer has a node for every transition and uses *softmax* activation.

## 3.1 Distributed Word Representations

POS-tag, FEATS and arc-label embeddings are initialized randomly and trained together with the network. Form embeddings are pre-trained using `word2vec` (Mikolov et al. [14]), employing the Skip-gram model with negative sampling.[3] We pre-train the embeddings only on the treebank data, to show that the resulting parser works with high accuracy without additional resources, which might be hard to obtain for some languages. Because all form embeddings are currently in

[3]The exact options for `word2vec` were the following: `-cbow 0 -size 50 -window 10 -negative 5 -hs 0 -sample 1e-1 -iter 15 -min-count 2`

the training data, we train them further together with the network, yielding a small accuracy improvement.

All forms appearing only once in the training data are replaced by a unique unknown-word token. Its embedding is then used for OOVs during parsing.

### 3.2 Training the Classifier

We train the neural network by stochastic gradient descent (Robbins and Monro [23]) with mini-batches of size 10, minimizing cross-entropy loss with $L_2$-regularization. We employ exponential learning-rate decay. For all treebanks, we use form embeddings of dimension 50, POS tag, FEATS and arc label embeddings of dimension 20, and a 200-node hidden layer. Other hyperparameters[4] are determined based on the development portion of the treebanks and the best combination is used.

We would like to note that although we tried several advanced neural network training techniques, notably AdaGrad (Duchi et al. [7]), dropout (Srivastava et al. [27]), cube activation function (reported to improve performance by Chen and Manning [2]), or AdaDelta (Zeiler [29]), none helped and the best accuracy was obtained by the basic mini-batched SGD.

### 3.3 Improving Classification Speed

We have used several techniques to improve the transition classification speed, which in turn directly determines parsing speed. Similar to Devlin et al. [6], we pre-computed the hidden layer increments for all embeddings and all input layer positions. We also compute the *tanh* using table lookup (except during training in order to obtain accurate gradients) and we do not normalize the output layer during parsing.

## 4 Search-based Oracle

When training the classifier using a static oracle, the same sequence of transitions is always used for every sentence. In other words, the classifier is trained only on transition sequences which do not contain any incorrect transitions. If the classifier is then used to parse a sentence and makes an error, it is difficult for it to recover from this error, because the classifier never encountered such situation in training data.

The dynamic oracle (Goldberg and Nivre [9]) improves the situation by being able to provide the best transition from an arbitrary configuration, even if some incorrect transitions have already been performed. When training with a dynamic oracle, usually an exploration policy parametrized by $k$ and $p$ is used to determine

[4] To be specific, the hyperparameters are: number of training iterations (between 5 and 10), initial learning rate (between 0.01 and 0.02), final learning rate (between 0.001 and 0.005) and $L_2$-regularization (between 0.1 and 0.5).

which transition to follow: during the first $k$ iterations the oracle transition is always chosen (as with the static oracle), but later the oracle transition is chosen only with probability $1-p$, using the (possibly incorrect) classifier prediction otherwise. Consequently, the classifier is being trained on sequences of transitions which it predicts itself.

The main idea behind our search-based oracle is to approximate the dynamic oracle by the current state of the classifier being trained. This approach is inspired by the Searn algorithm of Daumé III et al. [4], a method for reducing error propagation during structured prediction.

Specifically, when determining the transition to follow for a given parser configuration with the search-based oracle, we perform every applicable transition in sequence and for such transition we use the classifier being trained to parse the rest of the tree (by following the predicted transition in every step). We then choose such transition from the original configuration which results in a dependency tree with the highest attachment score.

As many transitions differ only in the label of the arc being added, to improve oracle speed, we employ the following heuristic: when choosing a transition to follow, we consider only those arc-adding transitions that assign the label appearing in the gold tree. This effectively reduces the number of possible transitions from tens to at most five (e.g., from 96 to 4 transitions in the `swap` system for English).

When training with the search-based oracle, we have to make sure that the original oracle is employed frequently enough, because the original oracle is the only way of utilizing gold data. Therefore, unlike with the dynamic oracle, where the exploration policy alternates between the dynamic oracle prediction and classifier prediction on every transition, we use the following policy: after training on *interval* sentences with the static oracle, we train one sentence with the search-based oracle. The *interval* becomes another hyperparameter of our system tuned on the development part of the treebank (we consider *interval* between 8 and 10).

The training time of a search-based oracle is naturally higher than the training time of a static oracle, because one prediction of a search-based oracle takes time linear in the size of the sentence being parsed. For the values of *interval* used, the training time of a search-based oracle is 2-3 times worse than training time of a static oracle alone. This is comparable to a dynamic oracle for the `stack` system, which is reported to have training time slower by a factor of 2.3 when using a dynamic oracle instead of a static one. Also note that this slowdown applies only to training, parsing speed of the trained classifier is exactly the same for static, search-based and dynamic oracles.

Interestingly, our search-based oracle can be combined not only with a static oracle, but also with a dynamic oracle, yielding accuracy improvements for the dynamic oracle, too.

# 5 Experiments

We evaluate parser accuracy on treebanks from the Universal Dependencies project, which seeks to develop cross-linguistically consistent treebank annotation for many languages. The annotation scheme is based on the universal Stanford dependencies (de Marneffe et al. [5]), the Google universal part-of-speech tags (Petrov et al. [22]), and the Interset interlingua for morphosyntactic features (Zeman [30]).

Namely, we use 37 dependency treebanks of Universal Dependencies 1.2 [21]. Four basic statistics of each treebank are presented in columns 2 and 3 of Table 1.

The results of our parser with the `stack`, `swap` and `arc2` systems are presented in the rest of Table 1. We report unlabeled attachment scores (UAS) and labeled attachment scores (LAS), excluding punctuation, computed using MaltEval (Nilsson and Nivre [15]). We show the results with a static oracle only and using our search-based oracle. For comparison, we also present results of a dynamic oracle (we implemented the dynamic oracle for the `stack` system from Goldberg et al. [11] and used it with the same classifier as the search-based oracle) and results of a search-based oracle used on top of a dynamic one.[5]

We also report results of MaltParser (Nivre et al. [19]), a greedy transition-based parser using liblinear (Fan et al. [8]) for optimization. We used MaltParser version 1.8.1. with default options and feature templates, changing the transition system (using `stackproj` and `stacklazy` as `stack` and `swap`, respectively), number of iterations (computed using treebank size), and passing a concatenation of UPOSTAG and FEATS fields as POS tags to use. We used MaltParser because it is a transition-based parser that implements many transition systems (including non-projective) which we wanted to compare with, and is very fast. It is therefore similar to our parser, in contrast to a slow parser achieving higher accuracy.

We also report parsing speed and model size of the `swap` parser. Parsing speed was measured on an Intel Pentium G850 2.9GHz CPU with 4GB RAM and it does not include model loading time.

## 5.1 Results

Comparing our static-oracle-only parser to MaltParser, our parser has better accuracy, achieving on average 6.2% relative error reduction in UAS and 6.7% in LAS. Our parser produces models on average half the size of MaltParser's (with models 4-5 times smaller for Czech, Ancient Greek, and Latin), and it is faster (20-30k words/s, on average 3.6 times faster than MaltParser).

The search-based oracle parser is clearly superior to the static oracle parser, achieving additional 4.3% relative error reduction in UAS and 3.6% relative error reduction in LAS.

---

[5] We did not implement any other dynamic oracle, because the dynamic oracle for the `arc2` system is very complicated with its $O(n^8)$ complexity, no dynamic oracle for the `swap` system is known to the best of our knowledge, and the recent dynamic oracle for the fully non-projective Covington parser of Gómez-Rodríguez and Fernández-González [12] uses a quite different transition system.

| Language | Size | Non-proj. | Static oracle | | | Search-based oracle | | | DynO | SB+DO | MaltParser | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Words | Non-proj. edges | Stack UAS | Swap UAS | Arc2 UAS | Stack UAS | Swap UAS | Arc2 UAS | Stack UAS | Stack UAS | Stack UAS | Swap UAS |
| | Sentences | Non-proj. sentences | Stack LAS | Swap LAS | Arc2 LAS | Stack LAS | Swap LAS | Arc2 LAS | Stack LAS | Stack LAS | Stack LAS | Swap LAS |
| Ancient Greek | 244 993 | 9.78% | 58.6 | 66.2 | 66.5 | 64.2 | **69.3** | 68.5 | 66.4 | 67.7 | *55.1* | *65.3* |
| | 16 221 | 63.22% | 53.0 | 60.6 | 60.9 | 58.5 | **63.9** | 62.8 | 60.5 | 62.0 | *49.4* | *59.4* |
| Ancient Greek–PROIEL | 206 966 | 5.95% | 72.3 | 75.7 | 74.8 | 74.4 | **76.1** | 75.5 | 75.8 | 75.9 | *69.7* | *73.4* |
| | 16 633 | 39.48% | 67.0 | 70.6 | 69.6 | 69.2 | **71.3** | 70.5 | 70.7 | 71.0 | *64.5* | *68.7* |
| Arabic | 282 384 | 0.33% | 79.9 | 79.8 | 80.2 | 80.4 | 80.6 | **80.7** | 78.2 | 79.4 | *80.1* | *79.7* |
| | 7 664 | 8.19% | 74.6 | 74.7 | 75.3 | 75.5 | **75.8** | 75.7 | 73.4 | 74.7 | *74.6* | *74.3* |
| Basque | 121 443 | 4.95% | 77.0 | 78.3 | 78.4 | 78.2 | 79.2 | 79.6 | 79.9 | **80.6** | *74.7* | *77.3* |
| | 8 993 | 33.74% | 71.9 | 73.1 | 73.2 | 73.5 | 74.3 | 74.5 | 75.2 | **76.0** | *68.9* | *71.5* |
| Bulgarian | 156 319 | 0.21% | 90.2 | 90.7 | 90.9 | 91.1 | 91.2 | **91.5** | 90.5 | 91.2 | *89.2* | *89.5* |
| | 11 138 | 2.83% | 84.8 | 85.5 | 85.7 | 86.0 | 86.1 | **86.2** | 85.3 | 86.0 | *83.2* | *83.6* |
| Croatian | 87 765 | 0.46% | 81.1 | 80.8 | 80.2 | 82.1 | 82.4 | 81.3 | **82.7** | 82.0 | *77.4* | *78.5* |
| | 3 957 | 7.48% | 73.9 | 73.6 | 72.5 | 75.2 | **75.3** | 74.4 | 74.8 | 74.7 | *69.7* | *70.9* |
| Czech | 1 506 490 | 0.93% | 86.7 | 87.9 | 87.8 | 87.7 | 88.0 | **88.2** | 87.2 | 87.5 | *85.2* | *86.3* |
| | 87 913 | 12.58% | 83.2 | 84.3 | 84.4 | 84.3 | 84.7 | **84.8** | 83.8 | 84.1 | *81.3* | *82.4* |
| Danish | 100 733 | 1.97% | 81.8 | 82.5 | 82.9 | 82.7 | 82.8 | **83.3** | 82.6 | **83.3** | *80.1* | *81.4* |
| | 5 512 | 22.84% | 78.0 | 79.1 | 79.3 | 79.2 | 79.2 | **80.0** | 78.8 | 79.6 | *75.5* | *76.8* |
| Dutch | 200 654 | 4.10% | 74.6 | 75.8 | 76.2 | 76.0 | **77.5** | 77.1 | 76.0 | 75.7 | *71.9* | *75.8* |
| | 13 735 | 30.87% | 70.8 | 72.0 | 71.8 | 72.0 | **73.8** | 73.1 | 72.1 | 72.3 | *67.9* | *71.2* |
| English | 254 830 | 0.48% | 86.7 | 86.5 | 86.9 | 87.4 | 87.2 | 87.3 | 87.3 | **87.7** | *86.3* | *86.5* |
| | 16 622 | 4.96% | 84.2 | 83.8 | 84.2 | **84.7** | 84.5 | 84.5 | 84.5 | **84.7** | *82.9* | *83.2* |
| Estonian | 9 491 | 0.08% | 85.0 | 85.3 | 86.0 | 87.4 | 86.5 | 86.3 | 86.4 | 86.2 | *86.4* | ***88.1*** |
| | 1 315 | 0.61% | 81.7 | 81.9 | 83.0 | 83.2 | 82.8 | 83.1 | 83.1 | 83.0 | *83.8* | ***85.7*** |
| Finnish | 181 022 | 0.74% | 80.4 | 81.2 | 81.1 | 81.5 | 81.7 | 81.6 | 82.7 | **83.5** | *81.0* | *80.8* |
| | 13 581 | 7.68% | 77.0 | 77.9 | 77.6 | 78.2 | 78.3 | 78.6 | 79.2 | **80.2** | *76.9* | *77.0* |
| Finnish–FTB | 159 829 | 1.09% | 80.3 | 80.1 | 80.0 | 81.3 | 81.0 | 80.4 | 81.6 | **82.3** | *79.6* | *80.1* |
| | 18 792 | 6.78% | 77.2 | 76.9 | 76.6 | 78.1 | 78.0 | 77.3 | 78.0 | **79.1** | *75.8* | *76.3* |
| French | 401 491 | 0.83% | 84.2 | 85.0 | 84.7 | 85.2 | **85.5** | 85.2 | 84.5 | 85.0 | *83.3* | *83.4* |
| | 16 446 | 12.45% | 80.4 | 81.2 | 81.1 | 81.5 | **81.7** | 81.4 | 80.6 | 81.2 | *78.8* | *78.8* |
| German | 298 242 | 0.90% | 82.3 | 82.6 | 83.0 | 83.3 | 83.3 | 83.1 | 83.2 | **84.4** | *81.3* | *82.2* |
| | 15 894 | 12.08% | 76.9 | 77.1 | 77.6 | 78.0 | 78.0 | 77.6 | 77.6 | **78.8** | *75.2* | *75.8* |
| Gothic | 56 128 | 3.86% | 76.2 | 76.1 | 76.2 | 78.3 | 77.4 | 77.9 | 78.0 | **78.5** | *75.2* | *76.2* |
| | 5 450 | 23.85% | 70.5 | 70.4 | 70.7 | 72.2 | 71.4 | 72.4 | 72.1 | **73.0** | *69.1* | *70.5* |
| Greek | 59 156 | 1.95% | 81.3 | 81.7 | 82.5 | **82.9** | 82.5 | **82.9** | 82.2 | 82.8 | *79.0* | *80.6* |
| | 2 411 | 27.87% | 78.4 | 78.4 | 79.2 | 79.3 | 79.1 | 79.6 | 79.0 | **79.8** | *75.2* | *77.1* |
| Hebrew | 158 855 | 0.00% | 85.1 | 86.0 | 85.9 | 86.0 | **86.2** | 86.1 | 85.6 | 85.8 | *83.2* | *83.1* |
| | 6 216 | 0.00% | 80.6 | 81.1 | 81.3 | 81.6 | **81.9** | 81.4 | 81.2 | 81.8 | *78.5* | *78.4* |
| Hindi | 351 704 | 0.76% | 92.5 | 93.3 | 93.0 | 93.3 | 93.7 | 93.6 | 93.8 | **93.9** | *89.4* | *89.5* |
| | 16 647 | 13.60% | 89.3 | 90.0 | 89.7 | 90.1 | 90.5 | 90.3 | **90.6** | **90.6** | *84.5* | *84.6* |
| Hungarian | 26 538 | 2.09% | 79.9 | 80.3 | 79.0 | 80.4 | 80.6 | 81.2 | 81.3 | **81.9** | *78.2* | *79.1* |
| | 1 299 | 25.17% | 74.2 | 74.3 | 72.9 | 75.1 | 75.5 | 75.6 | 75.8 | **77.5** | *72.7* | *74.0* |
| Indonesian | 121 923 | 0.13% | 83.1 | 83.1 | **83.3** | **83.3** | **83.3** | **83.3** | 82.1 | 82.4 | *81.7* | *81.8* |
| | 5 593 | 1.93% | 77.8 | 77.6 | 78.0 | 77.9 | **78.2** | 77.9 | 76.7 | 77.0 | *75.8* | *75.9* |
| Irish | 23 686 | 0.81% | 74.6 | 74.2 | 73.6 | 75.2 | 75.2 | 75.1 | 74.4 | 74.6 | ***75.4*** | *73.8* |
| | 1 020 | 12.84% | 67.4 | 66.8 | 66.7 | 68.1 | **68.5** | 67.5 | 68.0 | 67.7 | *67.6* | *66.4* |
| Italian | 271 180 | 0.32% | 90.1 | 90.0 | 90.3 | 90.6 | 90.6 | **90.8** | 89.8 | 90.6 | *89.0* | *88.8* |
| | 12 677 | 3.94% | 87.7 | 87.5 | 87.8 | 88.0 | 88.1 | **88.4** | 87.3 | 88.2 | *86.4* | *86.2* |
| Japanese–KTC | 267 631 | 0.00% | 85.1 | 85.2 | 84.9 | 85.5 | **85.7** | **85.7** | 85.1 | 85.3 | *84.2* | *84.1* |
| | 9 995 | 0.00% | 75.1 | 75.0 | 74.8 | **75.5** | 75.3 | 75.3 | 75.1 | 75.2 | *72.9* | *73.3* |
| Latin | 47 303 | 7.13% | 58.2 | 57.2 | 57.9 | 59.2 | 59.2 | 58.3 | **61.1** | 60.7 | *58.1* | *57.2* |
| | 3 269 | 46.22% | 49.8 | 50.4 | 50.6 | 51.7 | 52.0 | 51.0 | 53.6 | **53.9** | *50.2* | *50.1* |
| Latin–ITT | 259 684 | 3.45% | 77.2 | 80.5 | 79.0 | 77.8 | **80.8** | 79.3 | 79.8 | 79.5 | *72.4* | *76.3* |
| | 15 295 | 37.20% | 73.8 | 77.5 | 75.7 | 74.6 | **77.9** | 76.2 | 76.5 | 76.6 | *68.3* | *72.3* |
| Latin–PROIEL | 165 201 | 5.22% | 73.4 | 74.3 | 75.2 | 74.6 | 75.2 | 76.1 | 76.1 | **76.6** | *70.0* | *72.5* |
| | 14 982 | 30.09% | 68.3 | 69.3 | 70.1 | 69.5 | 70.3 | 71.0 | 70.8 | **71.5** | *64.8* | *67.7* |
| Norwegian | 311 277 | 0.60% | 89.2 | 89.2 | 89.7 | 89.8 | 90.0 | **90.1** | 89.7 | **90.1** | *88.9* | *88.9* |
| | 20 045 | 7.70% | 86.8 | 86.8 | 87.4 | 87.7 | 87.7 | **87.8** | 87.3 | **87.8** | *85.8* | *86.0* |
| Old Church Slavonic | 57 507 | 3.71% | 81.0 | 82.6 | 82.2 | 82.1 | **83.3** | 83.0 | 82.6 | 82.8 | *80.1* | *82.0* |
| | 6 346 | 21.57% | 75.4 | 77.8 | 76.9 | 77.0 | **78.0** | 77.9 | 77.5 | 77.9 | *75.0* | *77.2* |
| Persian | 152 871 | 0.38% | 83.8 | 83.1 | 83.5 | 84.5 | 84.2 | 84.6 | 84.8 | **85.0** | *80.8* | *80.8* |
| | 5 997 | 5.14% | 80.2 | 79.8 | 80.0 | 81.1 | 80.8 | 81.2 | 81.3 | **81.5** | *77.2* | *77.2* |
| Polish | 83 571 | 0.04% | 88.3 | 88.7 | 88.2 | 89.0 | 89.0 | 89.3 | **89.8** | 89.5 | *87.7* | *87.3* |
| | 8 227 | 0.32% | 84.1 | 84.6 | 83.8 | 84.8 | 84.5 | 85.2 | **85.5** | 85.2 | *83.1* | *82.8* |
| Portuguese | 212 545 | 1.27% | 85.8 | 87.6 | 87.5 | 87.5 | **88.4** | 88.1 | 86.9 | 87.5 | *84.5* | *85.5* |
| | 9 359 | 18.44% | 82.7 | 84.6 | 83.9 | 84.5 | **85.4** | 85.0 | 83.8 | 84.3 | *80.5* | *81.5* |
| Romanian | 12 094 | 0.89% | 75.4 | 74.5 | 76.3 | 76.7 | 76.9 | **77.4** | 75.5 | 76.3 | *72.8* | *73.1* |
| | 633 | 11.37% | 61.9 | 60.9 | 62.1 | 62.7 | **63.2** | **63.2** | 62.2 | 62.2 | *59.5* | *59.6* |
| Slovenian | 140 418 | 1.11% | 86.5 | 87.3 | 87.5 | 87.6 | **88.9** | 88.1 | 88.2 | 88.2 | *84.3* | *85.7* |
| | 7 996 | 13.61% | 84.5 | 85.4 | 85.4 | 85.8 | **87.0** | 86.0 | 86.1 | 86.4 | *81.9* | *83.4* |
| Spanish | 431 587 | 0.30% | 86.8 | 86.9 | 87.1 | **87.6** | 87.2 | 87.4 | 85.7 | 86.4 | *85.4* | *85.2* |
| | 16 013 | 6.05% | 83.6 | 83.7 | 83.7 | **84.4** | 84.1 | 84.0 | 82.5 | 83.4 | *81.2* | *81.2* |
| Swedish | 96 819 | 0.19% | 85.3 | 85.7 | 85.7 | 85.9 | 86.1 | 86.1 | **86.2** | **86.2** | *84.7* | *84.7* |
| | 6 026 | 2.77% | 81.4 | 81.9 | 82.0 | 82.3 | **82.5** | **82.5** | 82.4 | 82.4 | *80.3* | *80.5* |
| Tamil | 9 581 | 0.29% | 75.8 | 76.3 | 76.2 | 76.6 | 77.1 | 75.7 | **78.4** | 78.0 | *78.3* | *78.3* |
| | 600 | 2.17% | 67.1 | 68.5 | 67.5 | 67.9 | 68.7 | 67.3 | 69.6 | 69.5 | ***69.7*** | *69.4* |

Table 1: Parsing accuracy on all treebanks of Universal Dependencies version 1.2. *DynO* stands for dynamic oracle, *SB+DO* for search-based and dynamic oracle.

| Language | Size | | Swap system | | MaltParser | |
|---|---|---|---|---|---|---|
| | Words | Sentences | Speed kw/s | Model MB | Speed kw/s | Model MB |
| Ancient Greek | 244 993 | 16 221 | **27.7** | **3.9** | *9.5* | *23.2* |
| Ancient Greek–PROIEL | 206 966 | 16 633 | **25.9** | **3.4** | *8.7* | *21.2* |
| Arabic | 282 384 | 7 664 | **26.4** | **4.3** | *12.0* | *10.4* |
| Basque | 121 443 | 8 993 | **26.9** | **2.6** | *7.7* | *7.7* |
| Bulgarian | 156 319 | 11 138 | **27.5** | **3.2** | *10.6* | *6.8* |
| Croatian | 87 765 | 3 957 | **23.8** | **2.7** | *8.5* | *7.4* |
| Czech | 1 506 490 | 87 913 | **22.9** | **12.1** | *18.2* | *56.8* |
| Danish | 100 733 | 5 512 | **24.3** | **2.5** | *9.1* | *5.6* |
| Dutch | 200 654 | 13 735 | **26.4** | **3.2** | *11.8* | *9.2* |
| English | 254 830 | 16 622 | **21.8** | **3.2** | *12.5* | *6.3* |
| Estonian | 9 491 | 1 315 | **32.7** | 1.6 | *2.5* | ***0.8*** |
| Finnish | 181 022 | 13 581 | **22.9** | **4.1** | *9.5* | *14.5* |
| Finnish–FTB | 159 829 | 18 792 | **31.3** | **3.4** | *9.9* | *11.2* |
| French | 401 491 | 16 446 | **25.1** | 4.4 | *16.8* | ***4.1*** |
| German | 298 242 | 15 894 | **27.2** | **4.3** | *15.5* | *4.9* |
| Gothic | 56 128 | 5 450 | **27.6** | **2.0** | *6.7* | *6.0* |
| Greek | 59 156 | 2 411 | **28.3** | **2.1** | *6.4* | *4.5* |
| Hebrew | 158 855 | 6 216 | **22.7** | **2.9** | *11.3* | *8.1* |
| Hindi | 351 704 | 16 647 | **27.7** | **3.2** | *12.7* | *9.6* |
| Hungarian | 26 538 | 1 299 | **20.5** | **1.8** | *3.9* | *3.1* |
| Indonesian | 121 923 | 5 593 | **28.3** | **2.7** | *13.0* | *2.8* |
| Irish | 23 686 | 1 020 | **25.7** | **1.7** | *3.2* | *2.6* |
| Italian | 271 180 | 12 677 | **24.1** | **3.7** | *12.9* | *7.8* |
| Japanese–KTC | 267 631 | 9 995 | **29.3** | 1.4 | *17.7* | ***0.4*** |
| Latin | 47 303 | 3 269 | **28.5** | **2.1** | *5.7* | *7.3* |
| Latin–ITT | 259 684 | 15 295 | **26.7** | **2.6** | *11.3* | *15.8* |
| Latin–PROIEL | 165 201 | 14 982 | **25.7** | **3.1** | *8.0* | *18.2* |
| Norwegian | 311 277 | 20 045 | **25.9** | **3.6** | *12.9* | *7.6* |
| Old Church Slavonic | 57 507 | 6 346 | **28.1** | **2.1** | *6.6* | *5.6* |
| Persian | 152 871 | 5 997 | **25.2** | **2.7** | *12.2* | *3.9* |
| Polish | 83 571 | 8 227 | **30.2** | **2.5** | *8.2* | *6.3* |
| Portuguese | 212 545 | 9 359 | **27.4** | **3.4** | *12.4* | *8.2* |
| Romanian | 12 094 | 633 | **21.5** | **1.6** | *2.2* | *1.9* |
| Slovenian | 140 418 | 7 996 | **27.0** | **3.1** | *9.4* | *9.7* |
| Spanish | 431 587 | 16 013 | **26.9** | **4.8** | *13.6* | *12.0* |
| Swedish | 96 819 | 6 026 | **24.9** | **2.3** | *8.5* | *4.3* |
| Tamil | 9 581 | 600 | **31.1** | 1.6 | *2.3* | ***0.9*** |

Table 2: Parsing speed and model size measured on Universal Dependencies 1.2, using the `swap` transition system.

The dynamic oracle for the `stack` system has very similar results to the search-based oracle for the `stack` system (relative error reduction compared to static oracle is slightly higher for a search-based oracle than for a dynamic oracle), with the search-based oracle being simpler and applicable for any transition-based system. Additionally, the search-based oracle can be used together with the dynamic oracle, yielding further improvement of 2.2% relative error reduction in UAS and 2.3% relative error reduction in LAS on average over the UD 1.2 dataset.

## 6 Related Work

A neural network based dependency parser was proposed by Chen and Manning [2]. The architecture of our parser is quite similar. However, our parser implements two non-projective transition systems, it utilizes the search-based oracle, and we evaluate performance on 37 treebanks and without form embeddings computed on a large raw corpus.

Since parsing is a structured prediction problem, methods developed to handle error propagation during structured prediction like Searn (Daumé III et al. [4]), SMILe (Ross and Bagnell [24]) or DAgger (Ross et al. [25]) might improve parsing accuracy. The search-based oracle resembles Searn to some extent, as Searn computes the regret of an action by executing the current policy to gain a full sequence of predictions and computing its loss, which is similar to how optimal transitions in the search-based oracle are obtained. On the other hand, the rest of the training with the search-based oracle can be viewed as an approximation of the DAgger algorithm, similarly to the dynamic oracle (Goldberg and Nivre [9]).

Search-based oracle used with the `swap` transition system enables fully non-projective transition based parsing, for which no dynamic oracle existed for a long time. Recently, a dynamic oracle with $O(n)$ complexity for fully non-projective Covington parser was devised by Gómez-Rodríguez and Fernández-González [12]. The Covington parser can be implemented under the transition-based parsing framework (Nivre [17]), but it uses multiple lists of partially processed words and has quadratic worst-case complexity.

## 7 Conclusions

We have described a non-projective, neural-network based dependency parser Parsito[6] employing a novel, efficient search-based oracle. It has been evaluated on all 37 Universal Dependency treebanks, showing improvements in accuracy and especially in speed. We are releasing the parser and the models as open-source.[7]

The new search-based oracle improves parsing accuracy similarly to a dynamic one (over a static oracle), but it can work with the `swap` system for non-projective parsing (or any other transition system). Even when a polynomial-time dynamic oracle is known, the search-based oracle requires much less effort to implement, and there is still room for improvement (e.g., in the frequency of its use during training). Alternatively, the search-based oracle can be used together with the dynamic oracle to improve parsing accuracy even further.

Our future work includes utilizing character-level embeddings and/or computing word embeddings using large additional corpora. Furthermore, we will experiment with beam search for decoding as an option to improve parsing accuracy at the expense of parsing speed.

---

[6] Project homepage: `http://ufal.mff.cuni.cz/parsito`

[7] `http://hdl.handle.net/11234/1-1573`

## Acknowledgments

This work has been partially supported by and has been using language resources developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education of the Czech Republic (project LM2010013). This research was also partially supported by the SVV project number 260 224. We are grateful to the reviewers for comments which helped us to improve the paper.

## References

[1] Giuseppe Attardi. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 166–170, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[2] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[4] Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[5] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, 2014.

[6] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1370–1380, 2014.

[7] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12: 2121–2159, July 2011.

[8] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[9] Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 959–976, 2012.

[10] Yoav Goldberg and Joakim Nivre. Training deterministic parsers with non-deterministic oracles. *TACL*, 1:403–414, 2013.

[11] Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. A tabular method for dynamic oracles in transition-based parsing. *TACL*, 2:119–130, 2014.

[12] Carlos Gómez-Rodríguez and Daniel Fernández-González. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 256–261, 2015.

[13] Carlos Gómez-Rodríguez, Francesco Sartorio, and Giorgio Satta. A polynomial-time dynamic oracle for non-projective dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–927, Doha, Qatar, October 2014. Association for Computational Linguistics.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[15] Jens Nilsson and Joakim Nivre. Malteval: an evaluation and visualization tool for dependency parsing. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA).

[16] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160, 2003.

[17] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34(4):513–553, December 2008. doi: 10.1162/coli.07-056-R1-07-027.

[18] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 351–359, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[19] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *In Proceedings of LREC*, 2006. Available at `http://www.maltparser.org`.

[20] Joakim Nivre, Marco Kuhlmann, and Johan Hall. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 73–76, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[21] Joakim Nivre et al. Universal Dependencies 1.2, 2015. URL `http://hdl.handle.net/11234/1-1548`. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

[22] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[23] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951. doi: 10.1214/aoms/1177729586.

[24] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 661–668, 2010.

[25] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 627–635, 2011.

[26] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 455–465, 2013.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[28] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206, 2003.

[29] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[30] Daniel Zeman. Reusable tagset conversion using tagset drivers. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA).

[31] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 188–193, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.