Universal Morpho-syntactic Parsing and the Contribution of Lexica: Analyzing the ONLP Lab Submission to the CoNLL 2018 Shared Task

Amit Seker  
Open University of Israel  
amitse@openu.ac.il

Amir More  
Open University of Israel  
habeanf@gmail.com

Reut Tsarfaty  
Open University of Israel  
reutts@openu.ac.il

Abstract

We present the contribution of the ONLP lab at the Open University of Israel to the CoNLL 2018 UD SHARED TASK on MULTILINGUAL Parsing FROM RAW TEXT TO UNIVERSAL Dependencies. Our contribution is based on a transition-based parser called yap: yet another parser which includes a standalone morphological model, a standalone dependency model, and a joint morphosyntactic model. In the task we used yap’s standalone dependency parser to parse input morphologically disambiguated by UDpipe, and obtained the official score of 58.35 LAS. In a follow up investigation we use yap to show how the incorporation of morphological and lexical resources may improve the performance of end-to-end raw-to-dependencies parsing in the case of a morphologically-rich and low-resource language, Modern Hebrew. Our results on Hebrew underscore the importance of CoNLL-UL, a UD-compatible standard for accessing external lexical resources, for enhancing end-to-end UD parsing, in particular for morphologically rich and low-resource languages. We thus encourage the community to create, convert, or make available more such lexica.

1 Introduction

The Universal Dependencies (UD) initiative\footnote{universaldependencies.org} is an international, cross-linguistic and cross-cultural initiative aimed at providing morpho-syntactically annotated data sets for the world’s languages under a unified, harmonized, annotation scheme. The UD scheme (Nivre et al., 2016) adheres to two main principles: (i) there is a single set of POS tags, morphological properties, and dependency labels for all treebanks, and their annotation obeys a single set of annotation principles, and (ii) the text is represented in a two-level representation, clearly mapping the written space-delimited source tokens to the (morpho)syntactic words which participate in the dependency tree.

The CoNLL 2018 UD SHARED TASK is a multilingual parsing evaluation campaign wherein, contrary to previous shared tasks such as CoNLL-06/07 (Buchholz and Marsi, 2006; Nivre et al., 2007) corpora are provided with raw text, and the end goal is to provide a complete morpho-syntactic representation, including automatically resolving all of the token-word discrepancies. Contrary to the previous SPMRL shared tasks (Seddah et al., 2013, 2014), the output of all systems obeys a single annotation scheme, allowing for reliable cross-system and cross-language evaluation.

This paper presents the system submitted by the ONLP lab to the shared task, including the dependency models trained on the train sets, assuming morphologically disambiguated input tokens by UDpipe (Straka et al., 2016). We successfully parsed 81 test treebanks of UDv2 set (Nivre et al., 2017) participating in the CoNLL 2018 UD SHARED TASK (Zeman et al., 2018), obtaining the official score of LAS 58.35 average on all treebanks. We then present an analysis of case of Modern Hebrew, a low-resource morphologically rich language (MRL), which is known to be notoriously hard to parse, due to its high morphological word ambiguity and the small size of the treebank. We investigate the contribution of an external lexicon and a standalone morphological component, and show that inclusion of such lexica can lead to above 10% LAS improvement on this MRL.
Our investigation demonstrates the importance of sharing not only syntactic treebanks but also lexical resources among the UD community, and we propose the UD-compatible CoNLL-UL standard for external lexica (More et al., 2018) for sharing broad-coverage lexical resources in the next UD shared tasks, and in general.

The remainder of this document is organized as follows: In Section 2, we present our parser’s formal system and statistical models. In Section 3 we present technical issues relevant to the official run for the shared task followed by our results on all languages. In Section 4 we proceed with an analysis of the performance on Modern Hebrew in the task, compared against its performance augmented with a lexicon-backed morphological analyzer. We finally discuss in Section 5 directions for future work and conclude by embracing the importance, and a set of disambiguated morphemes $M$.

2 Our Framework

The parsing system presented by the ONLP Lab for this task is based on yap — yet another parser, a transition-based parsing system that relies on the formal framework of Zhang and Clark (2011), an efficient computational framework designed for structure prediction and based on the generalized perceptron for learning and beam search for decoding. This section briefly describes the formal settings and specific models available via yap.\(^2\)

2.1 Formal Settings

Formally, a transition system is a quadruple $(C, T, c_s, C_t)$ where $C$ is a set of configurations, $T$ a set of transitions between the elements of $C$, $c_s$ an initialization function, and $C_t \subset C$ a set of terminal configurations. A transition sequence $y = t_n(t_{n-1}(\ldots t_1(c_s(x))))$ for an input $x$ starts with an initial configuration $c_s(x)$ and results in a terminal configuration $c_n \in C_t$. In order to determine which transition $t \in T$ to apply given a configuration $c \in C$, we define a model that learns to predict the transition that would be chosen by an oracle function $O : C \to T$, which has access to the gold output. We employ an objective function

$$F(x) = \arg\max_{y \in \text{GEN}(x)} \text{Score}(y)$$

which scores output candidates (transition sequence in $\text{GEN}(x)$) such that the most plausible sequence of transitions is the one that most closely resembles the one generated by an oracle.

To compute $\text{Score}(y)$, $y$ is mapped to a global feature vector $\Phi(y) = \{\phi_i(y)\}$ where each feature $\phi_i(y)$ is a count of occurrences of a pattern defined by a feature function. Given this vector, $\text{Score}(y)$ is calculated as the dot product of $\Phi(y)$ and a weights vector $\vec{\omega}$:

$$\text{Score}(y) = \Phi(y) \cdot \vec{\omega} = \sum_{c_j \in y} \sum_i \omega_i \phi_i(c_j)$$

Following Zhang and Clark (2011), we learn the weights vector $\vec{\omega}$ via the generalized perceptron, using the early-update averaged variant of Collins and Roark (2004). For decoding, the framework uses the beam search algorithm, which helps mitigate otherwise irrecoverable errors in the transition sequence.

2.2 Morphological Analysis

The input to the morphological disambiguation (MD) component in particular and to the yap parsing system in general is a lattice $L$ representing all of the morphological analysis alternatives of $k$ surface tokens of the input stream $x = x_1, \ldots, x_k$, such that each $L_i = MA(x_i)$ is generated by a morphological analysis (MA) component, the lattice concatenate the lattices for the whole input sentence $x$. Each lattice-arc in $L$ has a morphosyntactic representation (MSR) defined as $m = (b, e, f, t, g)$, with $b$ and $e$ marking the start and end nodes of $m$ in $L$, $f$ a form, $t$ a universal part-of-speech tag, and $g$ a set of attributes=value universal features. These lattice-arc correspond to potential nodes in the intended dependency tree.

2.3 Morphological Disambiguation

The morphological disambiguation (MD) component of our parser is based on More and Tsarfaty (2016), modified to accommodate UD POS tags and morphological features. We provide here a brief exposition of the transition system, as shall be needed for our later analysis, and refer the reader to the original paper for an in-depth discussion (More and Tsarfaty, 2016).

A configuration for morphological disambiguation $C_{MD} = (L, n, i, M)$ consists of a lattice $L$, an index $n$ representing a node in $L$, an index $i$ s.t. $0 \leq i < k$ representing a specific token’s lattice, and a set of disambiguated morphemes $M$.
The initial configuration function is defined to be $c_s(x) = (L, \text{bottom}(L), 0, 0)$, where $L = MA(x_1) \circ \ldots \circ MA(x_k)$, and $n = \text{bottom}(L)$, the bottom of the lattice. A configuration is terminal when $n = \text{top}(L)$ and $i = k$. To traverse the lattice and disambiguate the input, we define an open set of transitions using the $MD_s$ transition template:

$$MD_s : (L, p, i, M) \rightarrow (L, q, i, M \cup \{m\})$$

Where $p = b, g = e$, and $s$ relates the transition to the disambiguated morpheme $m$ using a parameterized delexicalization $s = \text{DLEX}_oc(m)$:

$$\text{DLEX}_{oc}(m) = \begin{cases} (\ldots, t, g) & \text{if } t \in \text{OC} \\ (\ldots, f, t, g) & \text{otherwise} \end{cases}$$

In words, $\text{DLEX}$ projects a morpheme either with or without its form depending on whether or not the POS tag is an open-class with respect to the form. For UD, we define:

$$\text{OC} = \{\text{ADJ}, \text{AUX}, \text{ADV}, \text{PUNCT}, \text{NUM}, \ldots, \text{PROP}, \text{VERB} \}$$

We use the parametric model of More and Tsarfaty (2016) to score the transitions at each step. Since lattices may have paths of different length and we use beam search for decoding, the problem of variable-length transition sequences arises. We follow More and Tsarfaty (2016), using the ENDTOKEN transition to mitigate the biases induced by variable-length sequences.

### 2.4 Syntactic Disambiguation

A syntactic configuration is a triplet $C_{DEP} = (\sigma, \beta, A)$ where $\sigma$ is a stack, $\beta$ is a buffer, and $A$ a set of labeled arcs. For dependency parsing, we use a specific variant of Arc Eager that was first presented by (Zhang and Nivre, 2011). The differences between plain arc-eager and the arc-zeager variant are detailed in Figure 1.

The features defined for the parametric model also follows the definition of non-local features by Zhang and Nivre (2011), with one difference: we created one version of each feature with a morphological signature (all feature values of the relevant node) and one without, this allows to capture phenomena like agreement.

### 2.5 Joint Morpho-Syntactic Processing

Given the standalone morphological and syntactic disambiguation it is possible to embed the two into a single joint morpho-syntactic transition system with a “router” that decides which of the transition systems to apply in a given configuration, and train the morphosyntactic model to maximize a single objective function. We implement such joint parser in yap but we have not used it in the task, and we thus leave its description out of this exposition. For further discussion and experiments with the syntactic and joint morpho-syntactic variants in yap we refer the reader to (More et al., In Press).

### 3 Shared Task Implementation

For sentence segmentation and tokenization up to and including full morphological disambiguation for all languages, we rely on the UDPipe (Straka et al., 2016). Our parsing system implementation is yap – yet another parser, an open-source natural language processor written in Go\(^3\). Once compiled, the processor is a self-contained binary, without any dependencies on external libraries.

For the shared task the processor was compiled with Go version 1.10.3. During the test phase we wrapped the processor with a bash script that invoked yap serially on all the treebanks. Additionally, in order to train on all treebanks we limited the size of all training sets to the first 50,000 sentences for the parser.

Finally, our training algorithm iterates until convergence, where performance is measured by $F_1$ for labeled attachment score when evaluated on languages’ respective development sets. We define convergence as two consecutive iterations resulting in a monotonic decrease in $F_1$ for LAS, and used the best performing model up to that point. For some languages we observed the $F_1$ never monotonically decreased twice, so after 20 iterations we manually stopped training and used the best performing model.

For some treebanks (cs,cac, fr_sequoia, ru_syntagrus) the serialization code, which relies on Go’s built-in encoder package, failed to serialize the in-memory model because it is larger than 2\(^{30}\) bytes. To overcome the limitation we downloaded the go source code, manually changed the const field holding this limit and compiled the go source code.

Our strategy for parsing low resource languages was to use another treebank in the same language when such existed for the following:

\(^3\)https://golang.org
### Table 1: Parsing Results for Different Languages

| Language | CS-PDT | UD-Pipe | UD-Rank | DEP | DEP Rank |
|----------|--------|---------|---------|-----|----------|
| English  | 61.95  | 61.95   | 1       | 59.19 | 22       |
| Thai     | 21.11  | 21.11   | 22      | 21.11 | 22       |
| Arabic   | 21.11  | 21.11   | 22      | 21.11 | 22       |

Figure 1: Arc-Eager (Kübler et al., 2009, Chapter 3) and Arc-ZEager (Zhang and Nivre, 2011) Systems.

- **cs_pud**: cs_pdt
- **en_pud/en_lines**
- **fi_pud**: fi_fib
- **sv_pud**: sv_lines
- **ja_modern**: ja_gsd

For treebanks where no resource in the same language is available we used the parsing model trained for English:

- **br_keb**: en_ewt
- **fo_ofst**: en_ewt
- **pcm_nsc**: en_ewt
- **th_pud**: en_ewt

#### 4 The Case of MRLs: A Detailed Analysis for Modern Hebrew

As is well known, and as observed in this particular task, **morphologically rich languages** are most challenging to parse in the raw-to-dependencies parsing scenarios. This is because the initial automatic segmentation and morphological disambiguation may contain irrecoverable errors which will undermine parsing performance.

In order to investigate the errors of our parser we took a particular MRL that is known to be hard to parse (Modern Hebrew, ranked 58 in the LAS ranking, with baseline 58.73 accuracy) and contrasted the Baseline UDpipe results with the results of our parser, with and without the use of external lexical and morphological resources. Table 1 lists the results of the different parsing models on our dev set. In all of the parsing scenarios, we used UDpipe’s built-in sentence segmentation, to make sure we parse the exact same sentences. We then contrasted UDpipe’s full pipeline with the yap output for different morphological settings. We used the Hebrew UD train set for training and the Hebrew UD for analyzing the empirical results.

Initially, we parsed the dev set with the same system we used for the shared task, namely, yap dependency parser which parses the morphologically diambiguated output by UDpipe (yap DEP). Here we see that yap DEP results (59.19) are lower than the full UDpipe pipeline (61.95).
| Model         | Lexicon       | Sentence Segmentation | Morphological Disambiguation | Parser | Results on dev |
|--------------|---------------|-----------------------|-------------------------------|--------|----------------|
| UDpipe full  | –             | –                     | UDpipe                       | UDpipe | 61.95 / 49.28 / 51.45 |
| yap DEP      | –             | –                     | UDpipe                       | yap    | 59.19 / 49.19 / 33.75 |
| yap full     | Basline HebLex HebLex-Infused | UDpipe                   | yap                          | yap    | 52.25 / 37.85 / 29.59 |
|              |               |                       |                               |        | 60.94 / 39.49 / 33.85 |
|              |               |                       |                               |        | 71.39 / 61.42 / 41.86 |
| yap GOLD     | –             | Gold                  | Gold                         | yap    | 79.33 / 72.56 / 47.62 |

Table 1: The Contribution of Lexical Resources: Analysis of the Case for Modern Hebrew

We then moved on to experiment with yap’s complete pipeline, including a data-driven morphological analyzer (MA) to produce input lattices, transition-based morphological disambiguation and transition-based parsing. The results now dropped relative to the UDpipe baseline and relative to our own yap DEP system, from 59.19 to 52.25 LAS. Now, interestingly, when we replace the baseline data-driven MA learned from the treebank alone with an MA backed with an external broad-coverage lexicon called HebLex (adapted from (Adler and Elhadad, 2006)), the LAS results arrive at 60.94 LAS, outperforming the results obtained by yap DEP (UDpipe morphology with yap dependencies) and close much of the gap with the UDpipe full model. This suggests that much of the parser error stems from missing lexical knowledge concerning the morphologically rich and ambiguous word forms, rather than parser errors.

5 Discussion and Conclusion

This paper presents our submission to the CoNLL 2018 UD SHARED TASK. Our submitted system assumed UDpipe up to and including morphological disambiguation, and employed a state-of-the-art transition-based parsing model to successfully parse 81 languages in the UDv2 set, with the average LAS of 58.35, ranked 22 among the shared task participants.

A detailed post-task investigation of the performance that we conducted on Modern Hebrew, including the shared task and a number of variants, has shown that for the MRL case much of the parser errors may be attributed to incomplete morphological analyses or a complete lack thereof for the source tokens in the input stream.

In the future we intend to investigate sophisticated ways for incorporating additional external lexical and morphological knowledge, explicitly by means of broad-coverage lexica obeying the CoNLL-UL format (More et al., 2018), or implicitly by means of pre-trained word embeddings on large volumes of data. Note, however, that the utility of word-embedding themselves present an open questions in the case of morphologically rich and ambiguous source token, where each token may be equivalent to multiple syntactic words in a language like English.

We additionally intend to replace the handcrafted feature model with neural-network based feature extraction mechanisms, and we aim to explore universal morphosyntactic parsing via joint morphosyntactic modeling, as previously advocated in different settings (Tsarfaty and Goldberg, 2008; Bohnet and Nivre, 2012; Andor et al., 2016; Bohnet et al., 2013; Li et al., 2011; Bohnet and Nivre, 2012; Li et al., 2014; Zhang et al., 2014).
Acknowledgements

We thank the CoNLL Shared Task Organizing Committee for their hard work and their timely support. We also thank the TIRA platform team (Potthast et al., 2014) for providing a system that facilitates competition and reproducible research. The research towards this shared task submission has been supported by the European Research Council, ERC-StG-2015 Grant 677352, and by an Israel Science Foundation (ISF) Grant 1739/26, for which we are grateful.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-44, pages 665–672. https://doi.org/10.3115/1220175.1220259.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long
Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP-CoNLL ’12, pages 1455–1465. http://dl.acm.org/citation.cfm?id=2390948.2391114.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richard Farkas, Filip Ginter, and Jan Hajic. 2013. Joint morphological and syntactic analysis for richly inflected languages. *TACL* 1:415–428. http://dblp.uni-trier.de/db/journals/tacl/tacl1.html.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In Proceedings of CoNLL-X. pages 149–164.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL ’04. https://doi.org/10.3115/1218955.1218970.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency Parsing. Number 2 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, and Wenliang Chen. 2014. Joint optimization for chinese POS tagging and dependency parsing. *IEEE/ACM Trans. Audio, Speech & Language Processing* 22(1):274–286. https://doi.org/10.1109/TASLP.2013.2288081.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP ’11, pages 1180–1191. http://dl.acm.org/citation.cfm?id=2145432.2145557.

Amir More, Özlem Çetinoglu, Çagri Çöltekin, Nizar Habash, Benoît Sagot, Djamel Seddah, Dima Taji, and Reut Tsarfaty. 2018. Conll-ulf: Universal morphological lattices for universal dependency parsing. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.

Amir More, Amit Seker, Victoria Basanova, and Reut Tsarfaty. In Press. Joint transition-based models for morpho-syntactic parsing: Parsing strategies for mrls and a case study from modern hebrew. In *Transactions of ACL*.

Amir More and Reut Tsarfaty. 2016. Data-driven morphological analysis and disambiguation for morphologically rich languages and universal dependencies. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. The COLING 2016 Organizing Committee, Osaka, Japan, pages 337–348. http://aclweb.org/anthology/C16-1033.

Joakim Nivre, Željiko Agić, Lars Ahrenberg, et al. 2017. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-2184.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), European Language Resources Association, Portorož, Slovenia, pages 1659–1666.

Martin Potthast, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. Improving the reproducibility of PAN’s shared tasks: Plagiarism detection, author identification, and author profiling. In Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*. Springer, Berlin Heidelberg New York, pages 268–299. https://doi.org/10.1007/978-3-319-11382-1_22.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. pages 103–109.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, D. Jinho Choi, Richard Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Markus Kühnmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Secker, Yannick Versley, Veronika Vincze, Marcin Woźniak, Alina Wróblewska, and Ville-monte Eric de la Clergerie. 2013. Proceedings of the fourth workshop on statistical parsing of morphologically-rich languages. Association for Computational Linguistics.
tion for Computational Linguistics, pages 146–182. [http://aclweb.org/anthology/W13-4917.]

Milan Straka, Jan Hajič, and Jana Straková. 2016. UD-Pipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association, Portoro, Slovenia.

Reut Tsarfaty and Yoav Goldberg. 2008. Word-based or morpheme-based? annotation strategies for modern Hebrew clitics. In Proceedings of LREC.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. Association for Computational Linguistics, Brussels, Belgium, pages 1–20.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In In Proceedings of the ACL.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. Computational Linguistics 37(1):105–151. [https://doi.org/10.1162/coli_a_00037.]

Yue Zhang and Joakim Nivre. 2011. 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. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT ’11, pages 188–193. [http://dl.acm.org/citation.cfm?id=2002736.2002777.]

215