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[1] 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.

[1] universaldependencies.org

Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 208–215
Brussels, Belgium, October 31 – November 1, 2018.
https://doi.org/10.18653/v1/K18-2021

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 CoNLL-UL standard (More et al., 2018) for UD-anchored lexical resource as means to facilitate and improve raw-to-dependencies UD parsing.

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 $y = t_n(t_{n-1}(...t_1(c_s(x))))$ 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 $O : C \rightarrow T$, which has access to the gold output. We employ an objective function

$$F(x) = argmax_{y \in GEN(x)} Score(y)$$

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

To compute $Score(y)$, 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, $Score(y)$ is calculated as the dot product of $\Phi(y)$ and a weights vector $\vec{\omega}$:

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

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 $x = x_1, ..., x_k$, 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 *morpho-syntactic 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 attribute=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.

[2] https://github.com/habeanf/yap

The initial configuration function is defined to be $c_s(x) = (L, bottom(L), 0, \emptyset)$, where $L = MA(x_1) \circ ... \circ MA(x_k)$, and $n = bottom(L)$, the bottom of the lattice. A configuration is terminal when $n = top(L)$ *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 $p = b$, $q = e$, and s relates the transition to the disambiguated morpheme m using a parameterized delexicalization $s = DLEX_{oc}(m)$:

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

In words, $DLEX$ 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:

$$OC = \{^{ADJ,AUX,ADV,PUNCT,NUM,}_{INTJ,NOUN,PROPN,VERB}\}$$

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 σ is a stack, β 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

Arc Eager:	Conf.	$c = (\sigma, \beta, A)$	σ_h = *A second, 'head' stack*
	Initial	$c_s(x = x_1, ..., x_n) = ([0], [1, ..., n], \emptyset)$	
	Terminal	$C_t = \{c \in C \vert c = ([0], [], A)\}$	
	Transitions	$(\sigma, [i\vert\beta], A) \rightarrow ([\sigma\vert i], \beta, A)$	(SHIFT)
		$([\sigma\vert i], [j\vert\beta], A) \rightarrow ([\sigma\vert i\vert j], \beta, A \cup \{(i, l, j)\})$	(ArcRight$_l$)
		if $(k, l', i) \notin A$ *and* $i \neq 0$ *then*	
		$([\sigma\vert i], [j\vert\beta], A) \rightarrow ([\sigma], [j\vert\beta], A \cup \{(j, l, i)\})$	(ArcLeft$_l$)
		if $(k, l', i) \in A$ *then*	
		$([\sigma\vert i], \beta, A) \rightarrow (\sigma, \beta, A \cup \{(i, l, j)\})$	(REDUCE)

Arc ZEager:	Conf.	$c = (\sigma, \sigma_h, \beta, A)$	σ_h = *A second, 'head' stack*
	Initial	$c_s(x = x_1, ..., x_n) = ([]_\sigma, []_h, [1, ..., n], \emptyset)$	Note: no root
	Terminal	$C_t = \{c \in C \vert c = ([]_\sigma, \sigma_h, [], A)\}$	For any σ_h, A
	Transitions	$([i]_\sigma, \sigma_h, []_\beta, A) \rightarrow ([]_\sigma, \sigma_h, []_\beta, A)$	(POPROOT)
		$([\sigma\vert i], \sigma_h, []_\beta, A) \rightarrow (\sigma, \sigma_h, []_\beta, A)$	(REDUCE$_2$)
		if T_L! =REDUCE *then*	T_L = Last Transition
		$(\sigma, \sigma_h, [i\vert\beta], A) \rightarrow ([\sigma\vert i], [\sigma_h\vert i], \beta, A)$	(SHIFT)
		if $\vert\beta\vert > 0$ *and* ($\vert\sigma\vert > 1$ *and* ($\vert\beta\vert > 1$ *or* $\vert\sigma_h\vert = 1$)) *then*	
		$([\sigma\vert i], \sigma_h, [j\vert\beta], A) \rightarrow ([\sigma\vert i\vert j], \sigma_h, \beta, A \cup \{(i, l, j)\})$	(ArcRight$_l$)
		if $\vert\beta\vert > 0$ *and* $\vert\sigma\vert > 0$ *then*	
		$([\sigma\vert i], \sigma_h, \beta, A) \rightarrow ([\sigma], \sigma_h, \beta, A \cup \{(i, l, j)\})$	(REDUCE$_1$)
		if $\vert\beta\vert > 0$ *and* $\vert\sigma\vert > 0$ *and* $(k, l', i) \notin A$ *and* $i = k$ *then*	
		$([\sigma\vert i], [\sigma_h\vert k], [j\vert\beta], A) \rightarrow ([\sigma], [\sigma_h], [j\vert\beta], A \cup \{(j, l, i)\})$	(ArcLeft$_l$)

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_ftb
- 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_oft: en_ewt
- pcm_nsc: en_ewt
- th_pud: en_ewt

Tables 2 and 3 present our official results for all languages. Our system is ranked 22 with an average LAS score of 58.35. Our highest performing languages are Italian and Hindi — interestingly, both of which are considered morphologically rich, and both with LAS around 82. Our lowest performing languages (with up to 20 LAS) are the low-resource languages listed above, with Thai (0 LAS) as an outlier.

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 basline 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 LAS / MLAS / BLEX
UDPipe full	–	UDPipe	UDPipe	UDPipe	61.95 / 49.28 / 51.45
yap DEP	–	UDPipe	UDPipe	*yap*	59.19 / 49.19 / 33.75
yap full	Basline	UDPipe	*yap*	*yap*	52.25 / 37.85 / 29.59
	HebLex				60.94 / 39.49 / 33.85
	HebLex-Infused				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.

Finally, we simulated an ideal morphological lattices, by artificially infusing the path that indicates the correct disambiguation into the HebLex lattices in case it has been missing. Note that we still provide an ambiguous input signal, with many possible morphological analyses, only now we guarantee that the correct analysis exists in the lattice. For this setting, we see a significant improvement in LAS, obtaining 71.39 (much beyond the baseline) without changing any of the parsing algorithms involved. So, for morphologically rich and ambiguous languages it appears that lexical coverage is a major factor affecting task performance, especially in the resource scarce case.

Note that the upper-bound of our parser, when given a completely disambiguated morphological input stream, provides LAS of 79.33, which is a few points above the best system (Stanford) in the raw-to-dependencies scenario.

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, explicitely 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 ambigous source token, where each token may be equivalent to multiple syntactic words in a language like English.

We additionally intend to replace the hand-crafted 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)..

Treebank	UAS	LAS	MLAS	CLAS	BLEX
af_afribooms	79.83	73.01	60.73	64.99	41.46
ar_padt	71.62	64.87	53.54	60.69	5.13
bg_btb	87.94	77.81	69.16	73.11	25.18
br_keb	26.46	7.01	0.42	3.06	1.98
bxr_bdt	30.52	11.14	1.65	4.71	2.52
ca_ancora	86.94	80.49	70.37	72.92	53.34
cs_cac	85.41	76.25	64.77	73.22	24.89
cs_fictree	84.33	72.63	59.28	68.01	29.12
cs_pdt	83.09	74.6	65.51	72.12	29.79
cs_pud	74.26	49.25	31.59	35.74	15.69
cu_proiel	69.22	55.34	46.96	54.08	16.22
da_ddt	77.18	69.43	60.7	65.25	39.45
de_gsd	74.93	65.44	30.98	59.06	37.04
el_gdt	83.9	77.37	60.25	70.09	25.2
en_ewt	79.69	71	63.06	66.87	51.34
en_gum	78.24	69.91	58.8	62.85	46.55
en_lines	77.58	68.11	60.28	64.23	44.05
en_pud	72.12	63.73	53.73	58.79	43.33
es_ancora	87.61	81.55	73.01	75.48	52.31
et_edt	76.5	64.52	56.69	61.37	21.73
eu_bdt	69.27	55.21	43.35	50.71	21.14
fa_seraji	83.41	76.32	70.75	72.56	57.72
fi_ftb	75.5	59.82	48.98	52.84	16.82
fi_pud	56.32	38.3	40.59	44.04	15.81
fi_tdt	75.29	63.58	56.5	60.84	19.59
fo_oft	41.41	18.92	0.36	11.98	5.07
fr_gsd	84.89	78.15	69.18	72.91	50.51
fr_sequoia	83.7	77.93	69.18	72.92	49.55
fr_spoken	69.94	61.29	51.17	52.69	41.01
fro_srcmf	84.56	68.55	62.02	64.67	64.67
ga_idt	72.64	59.62	34.63	47.71	25.64
gl_ctg	80.39	75.59	63.47	68.3	40.79
gl_treegal	71.88	64.13	47.71	54.36	34.4
got_proiel	66.12	53.17	42.68	50.49	16.86
grc_perseus	46.75	35.21	17.91	28.08	5.24
grc_proiel	64.75	55.52	38.44	46.53	7.58
he_htb	63.19	55.94	43.12	46.99	31.79
hi_hdtb	91.1	82.35	65.28	77.12	59.25
hr_set	83.32	73.71	54.4	70.23	22.76
hsb_ufal	38.15	26.84	4.73	19.27	3.98
hu_szeged	68.53	54.32	41	48.46	30.73

Table 2: Official Shared-Task Results

Treebank	UAS	LAS	MLAS	CLAS	BLEX
hy_armtdp	40.83	25.04	8.42	17.09	6.96
id_gsd	82.19	72.23	61.93	70.05	41.53
it_isdt	89.25	82.22	72.45	75.06	52.7
it_postwita	73.97	66.35	53.96	56.95	41.03
ja_gsd	74.21	68.45	52.96	54.81	50.63
ja_modern	29.3	22.68	8.29	10.6	9.42
kk_ktb	37.92	17.34	4.33	9.25	2.89
kmr_mg	35.77	25.1	7.77	19.95	6.53
ko_gsd	69.4	55.13	49.99	51.85	17.26
ko_kaist	74.3	59.36	52.62	54.96	12.26
la_ittb	72.77	63.19	54.81	60.29	21.93
la_perseus	43.97	28.12	16.31	23.67	7.46
la_proiel	59.01	48.13	36.96	43.96	14.95
lv_lvtb	73.29	59.67	46.27	54.42	20.58
nl_alpino	79.22	70.95	56.57	62.7	42.48
nl_lassysmall	80.03	70.54	58.26	62.3	43.05
no_bokmaal	86.01	77.43	69.77	74.18	43.89
no_nynorsk	84.25	75.89	67.57	71.83	40.5
no_nynorsklia	58.15	42.21	32.58	36.57	30.99
pcm_nsc	26.11	12.4	4.57	14.68	14.68
pl_lfg	90.22	73.65	60.48	70.55	29.06
pl_sz	85.38	71.87	54.81	68.9	22.67
pt_bosque	84.35	77.46	62.64	69.99	52.06
ro_rrt	82.35	73.01	63.78	66.46	29.07
ru_syntagrus	82.93	74.39	66.49	71.54	23.83
ru_taiga	61.97	48.86	31.77	42.96	18.03
sk_snk	77.99	66.71	48.45	63.6	18.27
sl_ssj	78.61	71.24	57.51	67.57	23.79
sl_sst	51.43	40.88	28.96	35.62	22.96
sme_giella	64.09	48.62	37.23	44.08	17.25
sr_set	85.41	76.78	64.7	73.57	24.1
sv_lines	78.84	68.77	54.28	66.28	37.24
sv_pud	70.7	42.7	16.79	30.36	16.14
sv_talbanken	82.12	73.24	65.84	70.02	36
th_pud	0	0	0	0	0
tr_imst	60.24	43.95	34.37	39.01	15.95
ug_udt	67.47	45.67	28.89	35.67	24.12
uk_iu	78.45	69.36	51.42	64.69	21.97
ur_udtb	84.45	74.5	48.8	66.66	53.69
vi_vtb	45.43	37.05	32.18	33.78	33.78
zh_gsd	62.77	55.97	46.88	51.02	50.74

Table 3: Official Shared-Task Results

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*

Papers. http://aclweb.org/anthology/P/P16/P16-1231.pdf.

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, Richárd 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, Djamé Seddah, Dima Taji, and Reut Tsarfaty. 2018. Conll-ul: 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 Basmova, 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, Željko 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, Portoro, Slovenia, pages 1659–1666.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. pages 915–932.

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.

Djame Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, D. Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Villemonte Eric de la Clergerie. 2013. Proceedings of the fourth workshop on statistical parsing of morphologically-rich languages. Associa-

tion for Computational Linguistics, pages 146–182. http://aclweb.org/anthology/W13-4917.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: 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.