Universal Joint Morph-Syntactic Processing: The Open University of Israel's Submission to The CoNLL 2017 Shared Task

Amir More
Open University of Israel
habeanf@gmail.com

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

Abstract

We present the Open University's submission (ID OpenU-NLP-Lab) to the *CoNLL 2017 UD Shared Task* on multilingual parsing from raw text to Universal Dependencies. The core of our system is a joint morphological disambiguator and syntactic parser which accepts morphologically analyzed surface tokens as input and returns morphologically disambiguated dependency trees as output. Our parser requires a lattice as input, so we generate morphological analyses of surface tokens using a data-driven morphological analyzer that derives its lexicon from the UD training corpora, and we rely on UDPipe for sentence segmentation and surface-level tokenization. We report our official macro-average LAS is 56.56. Although our model is not as performant as many others, it does not make use of neural networks, therefore we do not rely on word embeddings or any other data source other than the corpora themselves. In addition, we show the utility of a lexicon-backed morphological analyzer for the MRL Modern Hebrew. We use our results on Modern Hebrew to argue that the UD community should define a UD-compatible standard for access to lexical resources, which we argue is crucial for MRLs and low resource languages in particular.

1 Introduction

The Universal Dependencies (UD) project (Nivre et al., 2016) sets itself apart from previous multilingual parsing initiatives such as the CoNLL (Buchholz and Marsi, 2006; Nivre et al., 2007) and SPMRL (Seddah et al., 2013, 2014) shared tasks with two key principles: (i) the POS tags, morphological properties, and dependency labels are unified, with enforceable annotation guidelines and (ii) corpora text is provided via a two-level representation of the input stream. With the latter two-level principle in place, corpora can be provided with raw text, syntactic words as the nodes of syntactic trees, and the relationship between them, in a harmonized scheme. This representation is crucial to the participation of Morphologically Rich Languages (MRLs) in end-to-end parsing tasks.

The availability of a wide range of language corpora in this manner provides a unique opportunity for the advancement of (universal) joint morpho-syntactic processing, introduced by Tsarfaty and Goldberg (2008) in a generative setting and advocated for in a variety of settings (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). To this end, our submission is a joint morpho-syntactic processor in a transition-based framework. We present our submission (OpenU-NLP-Lab), with models trained *only* on the train sets (Nivre et al., 2017b), parsing all 81 test treebanks of UD v2 corpora (Nivre et al., 2017a) participating in the *CoNLL 2017 UD Shared Task* (Zeman et al., 2017).

We use the results of our processor on an MRL to argue that one last piece of the puzzle is missing: a universal scheme for access to lexical resources. We discuss our results for a lexicon-backed approach, compared to a data-driven one. The goal of our submission is to compel the UD community to recognize the need for lexical resources in the context of joint morpho-syntactic processing, and push forward the discussion on a UD annotation-compliant standard for access to

Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 253–264,
Vancouver, Canada, August 3-4, 2017.

lexical resources that could benefit MRLs and low resource languages.

In section 2 we describe our framework and formal settings (2.1), first instantiated individually as a morphological disambiguator (2.2) and dependency parser (2.2), followed by how we unify the two into a joint processor (2.4).

Since the input stream of the processor is a morphological analysis of the tokenized raw text, we describe a universal, data-driven morphological analyzer, and a lexicon-based MA for the MRL Modern Hebrew (2.5).

In section 3, we detail the implementation of our parser (3.1) and specific technical issues we encountered with the official run for the shared task (3.2). We then present our results on all languages in section 4, and present a comparison to processing Modern Hebrew with a lexicon-based morphological analyzer. We discuss directions for future work in section 5, conclude with a summary of our submission in section 6, and urge the UD community to put forth a standard for lexical resource access.

2 Our Framework

We use the transition-based framework of Zhang and Clark (2011), originally designed for syntactic processing using the generalized perceptron and beam search, which we briefly cover in subsection 2.1.

We first describe the standalone transition system and model for morphological disambiguation of (More and Tsarfaty, 2016) (2.2), and Arc Standard transition system together with a rich-linguistic feature model (2.3). We then present our approach to joint morpho-syntactic processing which unifies both transition systems (2.4).

We present our baseline approach to data-driven morphological analysis, followed by our Modern Hebrew lexical resource (2.5).

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 the configuration $c_s(x)$. After n transitions of corresponding configurations $(t_i, c_i) \in T \times C$, the result is 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 need to 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 correct (gold) output.

To define a model, we employ an objective function $F : X \rightarrow \mathbb{R}$, which ranks outputs via a scoring of the possible transition sequences $(GEN(x))$ from which outputs are derived, such that the most plausible sequence of transitions is the one that most closely resembles one generated by an oracle:

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

How we define $Score$ is therefore crucial to the performance of the model, since it must capture the relation of a generated sequence (and its derived output) to that of an oracle's output. To compute $Score(y)$, y is mapped to a global feature vector $\Phi(y) = \{\phi_i(y)\}$ where each feature is a count of occurrences defined by feature functions. 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 Disambiguation

The morphological disambiguator (MD) component of our parser is based on More and Tsarfaty (2016), modified only to accommodate UD POS tags and morphological features. We provide a brief exposition of the transition system, and refer the reader to the original paper for an in-depth explanation (More and Tsarfaty, 2016).

The input to the transition-based MD is a lattice L of an input stream of k surface tokens $x = x_1, ..., x_k$, such that $L_i = MA(x_i)$, is generated by a morphological analysis component that analyzes each token separately and returns a lattice for the whole input sentence x. We rely on the UDPipe baseline models (Straka et al., 2016) for sentence segmentation and tokenization.

Each *lattice-arc* in L corresponds to a potential node in the intended dependency tree. A lattice-arc 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.

A configuration $C_{MD} = (L, n, i, M)$ 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 $c_s(x) = (L, bottom(L), 0, \emptyset)$, where $L = MA(x_1) \circ \ldots \circ MA(x_k)$, 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 redefine:

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

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.

2.3 Syntactic Disambiguation

For dependency parsing, we use the Arc Standard configuration, transition system, and oracle function defined in Kübler et al. (2009). A configuration is a triple $C_{DEP} = (\sigma, \beta, A)$ where σ is a stack, β is a buffer, and A a set of labeled arcs.

We present the specific variant of Arc Standard that we use in Figure 2.3. Note that in this variant, arc operations are performed between the top of the stack σ and the head of the buffer β. Additionally, in order to guarantee a single root, for the purposes of the shared task we apply a post processing step in which the first root node encountered (in left-to-right order) is designated as the only root node, and all other root nodes are set as its modifier with the "punct" dependency label.

Of course, this means that our transition system only applies to projective trees — the oracle will indeed fail given a non-projective tree, and our transition system cannot output one. In addition, since we are using the Arc Standard transition system, which has been shown to not be arc-decomposable, we cannot employ a dynamic oracle during training (Goldberg and Nivre, 2012).

The rich-linguistic feature model for our dependency parser, inspired by Zhang and Nivre (2011), applies the rich non-local features to arc standard (where this is possible), such as to accommodate the free word order of MRLs. We provide an appendix with a detailed comparison of the two feature models.

2.4 Joint Morpho-Syntactic Processing

Given standalone morphological and syntactic disambiguation systems in the same framework, we integrate them into a joint morpho-syntactic processor. Our integration is a literal embedding of the two systems, with a deterministic "router" that decides which of the two transition systems should apply a transition to a given configuration — we call this router a *strategy*.

We first must alter the morphological disambiguation transition such that a disambiguated morpheme is enqueued onto β:

$$\begin{aligned} MD_s : \quad & ((L, n, i, M), (\sigma, \beta, A)) \rightarrow \\ & ((L, q, j, M \cup \{m\}), (\sigma, [\beta|m], A)) \end{aligned}$$

We call the set of joint strategies used for the shared task $ArcGreedy_k$, because it will perform a syntactic operation if possible, otherwise it will disambiguate a morpheme. k determines the minimal number of morphemes in the buffer β of the Arc Standard configuration in order to perform a syntactic transition:

$$ArcGreedy_k(c_{md}, (\sigma, \beta, A)) = \begin{matrix} T_m & \textit{if } |\beta| \leq k \\ T_d & \textit{otherwise} \end{matrix}$$

Initial	$c_s(x = x_1, ..., x_n) = ([0], [1, ..., n], \emptyset)$	
Terminal	$C_t = \{c \in C \mid c = ([0], [], A)\}$	
Transitions	$(\sigma, [i\|\beta], A) \rightarrow ([\sigma\|i], \beta, A)$	(SHIFT)
	$([\sigma\|i], [j\|\beta], A) \rightarrow (\sigma, [j\|\beta], A \cup \{(j, l, i)\})$ *if* $i \neq 0$	(ArcLeft$_l$)
	$([\sigma\|i], [j\|\beta], A) \rightarrow (\sigma, [i\|\beta], A \cup \{(i, l, j)\})$	(ArcRight$_l$)

Figure 1: The Arc Standard transition system

We set $k = 3$ based on the features we use to predict the syntactic transition.

The $ArcGreedy$ approach provides joint processing through the interaction of the two systems through the global score. Together with beam search, this allows a syntactic transition to reverse the ranking of an otherwise higher-scored disambiguation candidate, and vice-versa, although this interaction occurs with a small delay due to the difference between a morphological disambiguation transition and a syntactic transition for the same morpheme.

2.5 Morphological Analysis

The joint parser requires a morphologically analyzed input, in the form of a lattice. However, universal lexical resources are not available for *all* languages participating in the shared task. Therefore, we use the data-driven morphological analyzer from More and Tsarfaty (2016), which derives its lexicon from the training set of a given UD corpora, modified to read/write UDv2-compatible file formats.

As part of our submission, we provide these derived lexica to the community.

In addition, we use the HEBLEX morphological analyzer from More and Tsarfaty (2016), adapted to output lattices conforming to UD annotation standards for universal POS tags and morphological features.

3 Implementation

In this section we describe technical details of implementation 3.1, bugs encountered during the shared task 3.2, and our approach to surprise languages 3.3.

3.1 Technical Details

For sentence segmentation and tokenization, we rely on the UDPipe (Straka et al., 2016) predicted data files. The morphological analysis component and joint morpho-syntactic parser are all implemented in yap[1] (yet another parser), an open-source natural language processor written in Go[2]. 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 1.8.1, and a git tag created for the commit used at the time of the task. During the test phase we wrapped the processor with a python script that invokes two instances concurrently in order to complete processing before the official (final) deadline.

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 *full morphological disambiguation* when evaluated on languages' respective development sets. We define convergence as two consecutive iterations resulting in a monotonic decrease in F_1 for full MD, 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.[3]

3.2 Shared Task Bugs

We encountered two serious bugs during training for the shared task, which prevented us from running our joint processor on all treebanks.

First, for some treebanks (cs_cac, cs_cltt, cs_pud, cs, en, 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. Much too our surprise, this is apparently an issue related to the decoder, one the Go maintainers are aware of but have decided not to address.[4] Changing our

[1] https://github.com/CoNLL-UD-2017/OpenU-NLP-Lab

[2] https://golang.org

[3] For PUD, we use models of "main" treebanks (no tcode)

[4] https://git.io/nogo

model serialization code was too large a task at the time we found it, so for the aforementioned problematic treebanks we had no choice but to train only the dependency parser, and rely on UDPipe for morphological disambiguation.

Second, close to the time of submitting this paper, we discovered a bug in the morphological disambiguator. The original MD model from More and Tsarfaty (2016) assumed the Hebrew treebank SPMRL annotation (SPMRL citation), in which some clitics are identified by morphological "suffix" features, as opposed to the UD approach which breaks them down as separate syntactic words. As a result, the MD transition system sometimes fails to *distinguish* between lattice-arcs.

As a temporary remedy, we modified the parser such that syntactic words with clitic suffixes have an additional indication as such, to set them apart from syntactic words without clitic suffixes. However, we did not have time to re-run our data-driven morphologically analyzed parses with this fix.

3.3 Surprise Languages

Our strategy for parsing surprise languages was to train a delexicalized (no word-form features) dependency-only parsing model on one treebank per surprise language, which we manually deemed as "close" as follows:

- bxr: ru_syntagrus
- kmr:fa
- sme: fi
- hsb: cs

We relied on the UDPipe predicted data up to and including full morphological disambiguation for all surprise languages.

4 Results and Discussion

In Tables 1 and 2 we present our official results for all languages. For the MRL Modern Hebrew, we train and test parsed using the lexicon-backed morphological analyzer (HEBLEX). When using HEBLEX, we obtained word-segmentation accuracy F_1 score of 87.48, compared to 81.26 in the data-driven MA of the official results, a 33% reduction in error rate.

Although the data-driven results suffer from the aforementioned bug, we do not expect them to change considerably, as we have seen such large differences with similar comparisons for the SPMRL Hebrew treebank. We hope the results from our unofficial run will be more convincing. It is important to note that the best word-segmentation result for Modern Hebrew in the shared task is 91.37.

We argue that although our lexicon-assisted model did not outperform the best model in the shared task, this does not invalidate our position on universal lexical resources. A 91.37 F_1 word-segmentation accuracy on Modern Hebrew is quite low, and in our opinion, still too low for inclusion in practical, real-world applications. We believe it is likely that together with access to lexical resources, more performant models would be able to bridge the gap and reduce this large error rate to a more acceptable level for down-stream tasks.

5 Future Work

In the future, we would like to replace our more traditional linear model with a modern, non-linear neural network-based approach. However, to date there is no solution for *joint* morph-syntactic processing of MRLs, a problem we aim to tackle. In the context of a neural-based solution, we believe that the availability of lexical resources will be crucial for MRLs and low resource languages in particular.

6 Conclusion

We present our submission to the *CoNLL 2017 UD Shared Task*, to the best of our knowledge the first universal, joint morpho-syntactic processor. We report our official result of 56.56. We contrast our results on the MRL Modern Hebrew, as a showcase of the utility of access to a lexicon-backed morphological analyzer.

Our goal is to instigate a discussion in the UD community on the need for a universal scheme for lexical resource access.

Acknowledgements

We would like to thank the CoNLL Shared Task Organizing Committee (COSTOCOM) for their hard word on the task and their timely support. We would also like to thank the TIRA platform team (Potthast et al., 2014) for providing a system that facilitates competition and reproducible research.

	UDPipe		yap				
Treebank	Sentences	Tokens	Words	UPOS	Feats	UAS	LAS
ar	84.57	99.98	92.48	82.73	21.13	53.41	45.01
ar_pud	100	80.89	89.68	65.49	33.49	41.54	31.84
bg	92.83	99.91	99.91	94.47	35.96	80.09	74.23
bxr	91.81	99.35	99.35	84.12	81.65	41.15	26.44
ca	98.95	99.97	99.78	93.55	19.48	80.09	74.53
cs*	92.03	99.9	99.9	98.13	91.01	81.45	76.44
cs_cac*	100	100	99.99	98.27	89.05	84.73	79.43
cs_cltt*	95.06	99.35	99.35	95.41	85.38	76.4	71.68
cs_pud*	96.43	99.29	99.29	96.55	87.34	81.33	75.75
cu	36.05	99.96	99.96	88.52	26.22	65.73	56.19
da	79.36	99.69	99.69	90.02	31.01	70.24	65.28
de	79.11	99.64	99.65	84.32	47.29	55.21	48.05
de_pud	86.49	97.97	97.7	77.61	30.13	52.98	44.05
el	90.79	99.88	99.88	92.29	29.89	77.8	73.41
en*	73.22	98.67	98.67	93.11	93.97	78.09	75.12
en_lines	85.84	99.94	99.94	91.78	99.94	73.5	68.09
en_partut	97.51	99.51	99.48	90.58	37.16	73.47	68.17
en_pud	97.13	99.66	99.66	89.6	32.92	78.27	73.47
es	94.15	99.87	99.42	91.14	42.85	73.51	67.89
es_ancora	97.05	99.97	99.72	93.73	16.99	76.74	71.34
es_pud	93.42	99.52	99.25	84.74	34.59	74.66	67.15
et	85.2	99.77	99.77	78.8	34.19	58.15	45.01
eu	99.58	99.96	99.96	87.06	37.02	66.24	56.37
fa	98	100	99.46	91.5	33.38	69.04	62.89
fi	84.56	99.63	99.63	84.99	29.62	57.65	45.99
fi_ftb	83.83	99.9	99.88	82.41	28.83	63.91	52.73
fi_pud	93.67	99.61	99.61	82.99	28.18	56.51	45.17
fr	93.59	99.75	99.49	92.54	42.33	77.28	71.96
fr_partut	98	99.83	99.44	93.61	34.13	78.84	73.1
fr_pud	92.32	99.1	98.79	84.73	36.8	73.68	67.67
fr_sequoia*	83.75	99.77	99.06	95.4	94.03	81.74	78.92
ga	95.81	99.29	99.29	83.69	28.66	67.69	54.53
gl	96.15	99.92	99.92	95.18	99.69	78.59	74.81
gl_treegal	81.63	99.59	98.02	86.8	22.08	66.68	59.77
got	27.85	100	100	89.19	27.67	59.25	50.06
grc	98.43	99.95	99.95	72.66	35.15	42.39	32.53
grc_proiel	43.11	100	100	89.8	25.26	59.13	51.05
he	99.39	99.94	81.26	73.55	31.71	46.67	41.49
hi	99.2	100	100	92.44	14.94	77.74	69.36
hi_pud	90.83	97.81	97.81	79.93	31.96	55.53	43.06
hr	96.92	99.93	99.93	89.45	19.84	68.49	59.94
hsb	90.69	99.84	99.84	90.3	74.02	64.5	57.14
hu	93.85	99.82	99.82	77.31	26.73	54.56	40.28
id	91.15	99.99	99.99	88.98	96.15	76.13	68.49
it	97.1	99.81	99.5	94.52	38.85	81.98	77.96
it_pud	96.58	99.59	99	88.37	33.98	80.02	75.11
ja	94.92	89.68	89.68	85.2	88.01	70.79	68.68
ja_pud	94.89	91.06	91.06	85.75	54.8	73.09	71.45

Table 1: Official results for the UD Shared Task. We include UDPipe predicted measures for completeness. Our system does not predict lemmas and XPOS, so we do not show them. Treebanks with * were processed by only our dependency parser, relying on UDPipe for morphological disambiguation, due to a technical issue.

	UDPipe		yap				
Treebank	Sentences	Tokens	Words	UPOS	Feats	UAS	LAS
kk	81.38	95.2	94.9	43.87	33.45	36.12	10.49
kmr	97.02	99.01	98.85	90.04	80.72	51.24	38.61
ko	93.05	99.73	99.73	81.9	98.99	60.75	52.37
la	98.09	99.99	99.99	68.32	36.76	37.04	24.3
la_ittb	93.24	99.99	99.99	93.65	31.57	57.99	50.65
la_proiel	25.8	100	100	87.32	26.48	46.46	37.12
lv	98.59	98.91	98.91	80.81	39.93	58.13	47.71
nl	77.14	99.88	99.88	80.07	7.56	50.27	41.9
nl_lassysmall	78.62	99.93	99.93	95.09	47.01	73.8	69.78
no_bokmaal	95.76	99.75	99.75	93.4	40.05	80.68	76.69
no_nynorsk	91.23	99.85	99.85	92.69	40.01	76.51	71.89
pl	98.91	99.99	98.97	86.81	26.09	71.07	63.03
pt	89.79	99.64	99.27	88.48	35.63	59.62	53.67
pt_br	96.84	99.94	99.84	94.44	90.12	83.9	80.65
pt_pud	95.65	99.29	99.2	82.17	37.21	60.51	53.87
ro	93.42	99.64	99.64	93.86	16.29	79.55	72.14
ru	96.42	99.91	99.91	84.32	37.04	64.02	57.09
ru_pud	98.95	97.18	97.18	75.21	35.31	61.06	52.14
ru_syntagrus*	97.81	99.57	99.57	97.99	93.47	84.2	80.1
sk	83.53	100	100	77.99	19.11	57.95	50.25
sl	99.24	99.96	99.96	89.19	23.64	70.95	65.27
sl_sst	16.72	99.82	99.82	83.24	29.7	45.76	37.11
sme	98.79	99.88	99.88	86.81	81.25	45.03	32.57
sv	96.37	99.84	99.84	92.23	34.17	75.96	70.38
sv_lines	86.44	99.98	99.98	91.39	99.98	75.1	69.21
sv_pud	90.2	98.26	98.26	82.66	32.27	68.35	61.42
tr	96.63	99.85	97.17	83.15	35.75	52.72	39.82
tr_pud	93.91	98.86	95.7	61.52	23.36	44.73	23.95
ug	63.55	98.52	98.52	66.15	98.52	20.7	7.35
uk	92.59	99.81	99.81	76.27	30.24	53.24	42.29
ur	98.32	100	100	88.58	14.53	77.95	69.89
vi	92.59	82.47	82.47	71.84	78.23	41.25	36.51
zh	98.19	88.91	88.91	80.08	78.27	58.03	52.93

Table 2: Official results for the UD Shared Task. We include UDPipe predicted measures for completeness. Our system does not predict lemmas and XPOS, so we do not show them. Treebanks with * were processed by only our dependency parser, relying on UDPipe for morphological disambiguation, due to a technical issue.

Appendix

Dependency Features

We use the feature description scheme of Zhang and Nivre (2011) for easy comparison.

Let $c = (S, N, A)$ be a configuration where S is the stack, N is the buffer.

We define an *address* as the location of a node in the partial dependencies trees in S and N of configuration c. An address has a structure name S or N, a subscript integer to access a k-deep node, and characters to access the heads or dependents of the node found at S_k or N_k. For example, the address S_{0h} refers to the head (if such exists) of the partial tree found at the top of the stack. The address N_1 refers to the node that is second in the buffer.

Rich Linguistic Feature Types

In addition to features described in Zhang and Nivre (2011), we define the following attributes:

- f_p - the multi-set of parts of speech of the dependents of a node
- s_f - the multi-set of labels of all dependents of a node
- v_f - the valency (= number) of all dependents of a node

Also, we define C_i as an address generator - it generate a feature for each dependent of the addressed node.

Morphological Augmentation

To allow the inclusion of morphology we add the ability of specifying morphological properties to be added to all features of a feature group. Augmentation of a feature *group* does not cause a replacement of the defined features, it only creates a copy with the addition of morphological properties.

To augment a feature group, all the features to the groups are required to have the same number of addresses. An augmentation specifies a character, either h or x, to specify the host or suffix morphological properties as attributes, respectively. If the group has more than one address, the augmentation must specify an address (a 1-indexed integer offset). Multiple augmentations may be used together.

For example, given the feature group Pairs in table 3, the first few features are $S_w t N_0 wt$, $S_0 wt N_0 w$, $S_w N_0 wt$, etc. All features in the Pairs group have two addresses. An example of a morphological augmentation of the Pairs group is $h1h2$, resulting in the new features $S_0 wtm_h N_0 wtm_h$, $S_0 wtm_h N_0 wm_h$, $S_w m_h N_0 wtm_h$, etc. where m_h is the set of key-value pairs of properties of the respective morphemes at the top of the stack (S_0) and buffer (N_0).

Features

The set of rich non-local features of (Zhang and Nivre, 2011) and the new rich linguistic features defined in this work are shown in table 3. The features are shown side by side to ease the comparison of the two feature sets, along with a column indicating the changes made.

The feature groups are augmented with morphological properties as defined in table 6.

N-L Group	N-L Feature	Ling. Feature	Ling. Group	Change
Single	S_0w	S_0w	Single	
Single	S_0t	S_0t	Single	
Single	S_0wt	S_0wt	Single	
Single	N_0w	N_0w	Single	
Single	N_0t	N_0t	Single	
Single	N_0wt	N_0wt	Single	
Single	N_1w	N_1w	Single	
Single	N_1t	N_1t	Single	
Single	N_1wt	N_1wt	Single	
Single	N_2w	N_2w	Single	
Single	N_2t	N_2t	Single	
Single	N_2wt	N_2wt	Single	
Pairs	S_0wtN_0wt	S_0wtN_0wt	Pairs	
Pairs	S_0wtN_0w	S_0wtN_0w	Pairs	
Pairs	S_0wN_0wt	S_0wN_0wt	Pairs	
Pairs	S_0wtN_0t	S_0wtN_0t	Pairs	
Pairs	S_0tN_0wt	S_0tN_0wt	Pairs	
Pairs	S_0wN_0w	S_0wN_0w	Pairs	
Pairs	S_0tN_0t	S_0tN_0t	Pairs	
Pairs	N_0tN_1t	N_0tN_1t	Pairs	
Three Words	$N_0tN_1tN_2t$	$N_0tN_1tN_2t$	Three Words (A)	
Three Words	$S_0tN_0tN_1t$	$S_0tN_0tN_1t$	Three Words (A)	
Three Words	$S_{0h}tS_0tN_0t$	$S_{0h}tS_0tN_0t$	Three Words (A)	
Three Words	$S_0tN_0tN_{0ld}t$	$S_0tN_0tf_p$	Three Words (B)	$N_{0ld}t \rightarrow N_0f_p$
Three Words	$S_0tS_{0ld}tN_0t$	$S_0tf_pN_0t$	Three Words (B)	$ld/rd \rightarrow f_p$
Three Words	$S_0tS_{0rd}tN_0t$		Three Words (B)	
Distance	S_0wd	S_0wd	Distance	
Distance	S_0td	S_0td	Distance	
Distance	N_0wd	N_0wd	Distance	
Distance	N_0td	N_0td	Distance	
Distance	S_0wN_0wd	S_0wN_0wd	Distance	
Distance	S_0tN_0td	S_0tN_0td	Distance	
Valency	S_0wv_r	S_0wv_f	Valency frames	
Valency	S_0wv_l		Valency frames	
Valency	S_0tv_r	S_0tv_f	Valency frames	$v_r/v_l \rightarrow v_f$
Valency	S_0tv_l		Valency frames	
Valency	N_0wv_l	N_0wv_f	Valency frames	
Valency	N_0tv_l	N_0tv_f	Valency frames	
Unigrams	$S_{0h}w$	$S_{0h}w$	Unigrams (A)	
Unigrams	$S_{0h}t$	$S_{0h}t$	Unigrams (A)	
Unigrams	S_0l	S_0l	Unigrams (A)	
Unigrams	$S_{0ld}w$	$S_0wS_0C_iw$	Unigrams (B)	
Unigrams	$S_{0ld}t$	$S_0wS_0C_it$	Unigrams (B)	
Unigrams	$S_{0ld}l$	$S_0wS_0C_il$	Unigrams (B)	Switch to non-directional bi-lexical dependencies, C_i = for each dependent
Unigrams	$S_{0rd}w$	$S_0tS_0C_iw$	Unigrams (B)	
Unigrams	$S_{0rd}t$	$S_0tS_0C_it$	Unigrams (B)	
Unigrams	$S_{0rd}l$	$S_0tS_0C_il$	Unigrams (B)	
Unigrams	$N_{0ld}w$	$N_0wN_0C_iw$	Unigrams (B)	
Unigrams	$N_{0ld}t$	$N_0wN_0C_it$	Unigrams (B)	
Unigrams	$N_{0ld}l$	$N_0wN_0C_il$	Unigrams (B)	
Unigrams		$N_0tN_0C_iw$	Unigrams	
Unigrams		$N_0tN_0C_it$	Unigrams	New
Unigrams		$N_0tN_0C_il$	Unigrams	
Third Order	$S_{0l2d}w$		Third Order	
Third Order	$S_{0l2d}t$		Third Order	
Third Order	$S_{0l2d}l$		Third Order	
Third Order	$S_{0r2d}w$		Third Order	
Third Order	$S_{0r2d}t$		Third Order	Removed
Third Order	$S_{0r2d}l$		Third Order	
Third Order	$N_{0l2d}w$		Third Order	
Third Order	$N_{0l2d}t$		Third Order	
Third Order	$N_{0l2d}l$		Third Order	
Third Order	$N_0tN_{0ld}tN_{0l2d}t$	N_0tf_p	Third Order	$N_{0l2d}t \rightarrow N_0f_p$
Third Order	$S_{0h2}w$	$S_{0h2}w$	Third Order (A)	
Third Order	$S_{0h2}t$	$S_{0h2}t$	Third Order (A)	
Third Order	$S_{0h}l$	$S_{0h}l$	Third Order (A)	
Third Order	$S_0tS_{0ld}tS_{0l2d}t$	S_0tf_p	Third Order (B)	$ld/rd/l2d/r2d \rightarrow f_p$
Third Order	$S_0tS_{0rd}tS_{0r2d}t$		Third Order (B)	
Third Order	$S_{0h2}tS_{0h}tS_0t$	$S_{0h2}tS_{0h}tS_0t$	Third Order (C)	
LabelSet	S_0wl_p	S_0ws_f	Subcat. frames	
LabelSet	S_0wr_p		Subcat. frames	
LabelSet	S_0tr_p	S_0ws_f	Subcat. frames	$l_p/r_p \rightarrow s_f$
LabelSet	S_0tl_p		Subcat. frames	
LabelSet	N_0wl_p	N_0ws_f	Subcat. frames	
LabelSet	N_0tl_p	N_0ts_f	Subcat. frames	
		S_0wS_0o	Edge Potential	New
		S_0tS_0o	Edge Potential	$o = \|\sigma_h\|$ = edge potential
		N_0tS_0o	Edge Potential	
		N_0wS_0o	Edge Potential	

Table 3: Rich Non-Local Features vs. Rich Linguistic Features

Feature Group	Morphological Augmentations
Single	h x
Pairs	h1h2 h1x2 x1h2
Three Words (A)	h1h2 h1x2 x1h2 h1h3 h1x3 x1h3 h2h3 h2x3 x2h3
Three Words (B)	h1h3 h1x3 x1h3
Valency	h
Unigram (A)	h x
Bigram	h1h2 h1x2 x1h2
Third Order (A)	h x
Third Order (B)	h x
Third Order (C)	h1h2 h1x2 x1h2 h1h3 h1x3 x1h3 h2h3 h2x3 x2h3

Table 4: Morphological Augmentation of Rich Linguistic Feature Groups

References

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.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*. pages 959–976. http://aclweb.org/anthology/C/C12/C12-1059.pdf.

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 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. 2017a. 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.

Joakim Nivre et al. 2017b. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, `http://hdl.handle.net/11234/1-1983`. http://hdl.handle.net/11234/1-1983.

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. Association 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, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökırmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Hěctor Martínez Alonso, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

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.