Multi-Model and Crosslingual Dependency Analysis

Johannes Heinecke, Munshi Asadullah
Orange Labs
2 avenue Pierre Marzin
F - 22300 Lannion, France
{johannes.heinecke,munshi.asadullah}@orange.com

Abstract

This paper describes the system of the team Orange-Deskiñ, used for the *CoNLL 2017 UD Shared Task*. We based our approach on an existing open source tool (BistParser), which we modified in order to produce the required output. Additionally we added a kind of pseudo-projectivisation. This was needed since some of the task's languages have a high percentage of non-projective dependency trees. In most cases we also employed word embeddings. For the 4 surprise languages, the data provided seemed too little to train on. Thus we decided to use the training data of typologically close languages instead. Our system achieved a macro-averaged LAS of 68.61% (10th in the overall ranking) which improved to 69.38% after bug fixes.

1 Introduction

For our work in our lab (Orange-Deskiñ) we needed a robust dependency analysis for written French with the highest Labeled Attachment Score (LAS)[1] possible, using a wide range of dependency relations. Having worked in the past on rule based dependency analysis, it became obvious that we need to adopt a more modern approach to dependency analysis. Thus during the last year we tried several freely available open source tools available (e.g. MaltParser[2], Google's SyntaxNet[3], Standford Dependency Tools[4], BistParser[5] and HTParser[6]), trained on different Treebanks (notably French Sequoia (Candito et al., 2014) and Universal Dependencies (McDonald et al., 2013)). All combinations of tools and treebanks had some advantages and some inconveniences. For instance, the underlying linguistic models of the treebanks are not the same or some tools would not accept CONLLU input but only raw text and apply their own segmentation and POS tagging.

In a next step we enriched the French treebanks with additional information like lemmas, morphological features and more fine-graded XPOS in addition to the about 20 UPOS categories of the treebanks (UD-French v1.2 does not contain neither lemmas nor morphological features) and conducted a new training/test/evaluation cycle. Since the initial results for French were encouraging we tried the same approaches with other languages, such as the languages proposed for *CoNLL 2017 UD Shared Task* (Zeman et al., 2017). However, for participation at the shared task, we relied exclusively on the data provided by Universal Dependencies (Nivre et al., 2016, 2017b), also for French in spite of our previous work.

For the shared task we have trained models separately for each language. So strictly speaking, this is not a multilingual but a monolingual multi-model approach.

2 System Description

2.1 Software

For the shared task, we used an (older) version of BistParser for all treebanks (ud-treebanks-conll2017). BistParser (Kiperwasser and Goldberg, 2016) is a transition based parser (Nivre (2008), and which uses the arc-hybrid transition

[1] Since we are interested in semantic relations a good CLAS score (Nivre and Fang, 2017) is even more relevant.
[2] http://www.maltparser.org/
[3] https://www.tensorflow.org/versions/r0.11/tutorials/syntaxnet/
[4] https://nlp.stanford.edu/software/stanford-dependencies.shtml
[5] https://github.com/elikip/bist-parser
[6] https://github.com/elikip/htparser

system (Kuhlmann et al., 2011)) with the three "basic" transitions LEFT ARC, RIGHT ARC and SHIFT. Since the shared task requires that output dependency trees have exactly one root, we modified BistParser accordingly by deleting the additional ROOT node added to each sentence in the original version of this parser. BistParser uses a bidirectional LSTM neural network. Currently BistParser uses forms and XPOS for both learning and predicting. We have started implementing the use of feature column as well, but this has not been used for the *CoNLL 2017 UD Shared Task*.

Some of the languages in the shared task have a large percentage of non-projective sentences. We thus decided to implement a pseudo-projectivisation (Kübler et al., 2009, p. 37) of the input sentences before training or predicting. The output sentences are than de-projectivised. Sometimes of course, the de-projectivisation can fail, especially if there are other dependency relation errors. Our tests showed, however, that the overall result for most languages is still better than without any pseudo-projectivisation.

Finally we implemented filters which ignore the special CONLLU lines for multi-word tokens (`2-3 ...`) and elliptic insertions (`4.1 ...`) and reinsert those lines after predicting.

In order to reduce memory usage during training and prediction, we modified BistParser and the underlying CNN library[7] to load word embeddings only for the words present in the training or test data. For the same reason we modified BistParser to read sentences one by one, to predict, and to output the result, instead of reading the entire test file at once[8].

2.2 Training Data

We trained our models using all treebanks provided by the *CoNLL 2017 UD Shared Task*. Since for some of the languages there were no development treebanks available, we split the training treebank in order to get a small development corpus (10% of the training corpus is split to test during development). This posed a certain problem for treebanks like Kazakh and Uyghur, which are hopelessly small (31 and 100 sentences respectively). Eventhough both languages are genetically and typologically very close to Turkish (3685 sentences), we finally trained on those small treebanks for time constraints (with more time available we would have experimented with various other parameters and a cross-lingual approach).

In most cases, adding word embeddings improved the LAS considerably. We downloaded the language specific corpora provided[9] by the task organisers and calculated our own word embeddings with Mikolov's `word2vec` (Mikolov et al., 2013)[10], which gave better results than the 100-dimensional word embeddings provided. In order to get the best results, we cleaned the text corpora (e.g. deleting letter-digit combinations and separating punctuation symbols such as commas, question marks etc. by a white space from the preceding token). For those languages which use an alphabet which has case distinction (Latin, Cyrillic and Greek) we put everything in lowercase. Finally we trained word embeddings with 300 and 500 dimensional vectors respectively. For all other parameters of `word2vec` we used the default setting, apart from the lower frequency limit, which we increased to 15 words.

The word embeddings were calculated on a server with a 32 core CPU running Ubuntu 14.04[11]. For the biggest text corpora like English (9 billion words), German (5,9 billion words), Indonesian (5 billion words) French (4,8 billion words) training for 500 dimensional word vectors took up to 6 hours (English).

A similar approach to `word2vec` is fastText The fundamental difference is the adoption of the "subword model" described in Bojanowski et al. (2016). A subword model is described as a open model allowing each word to be represented not only by the word itself but also the subword components of the word in combination. Subword components can be n-grams with varying values for n, stems, root words, prefixes, and suffixes or any other possible formalism. As a matter of fact, `word2vec` can been seen as the minimum configuration of fastText where only the words are considered. FastText has been demonstrated (Joulin et al., 2016) to perform rather well in two different tasks i.e. sentiment analysis and tag prediction. For the *CoNLL 2017 UD Shared Task* we finally

[7] https://github.com/clab/cnn-v1, CNN has meanwhile evolved to Dynet (https://github.com/clab/dynet)

[8] The code is available at https://github.com/CoNLL-UD-2017/Orange-Deskin

[9] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989

[10] https://code.google.com/archive/p/word2vec/

[11] Intel Xeon CPU E5-2640 v3 at 2.60GHz.

used `word2vec`, since the results were similar, but fastText was taking significantly more time to train.

3 Training and development

We trained all treebanks without any word embeddings, with 300 and with 500 dimensional word embeddings. For BistParser, the only other parameter we changed was the size of the first hidden layer (default 100) which we set to 50 (or lower, especially for languages whose treebanks are very small). Every sentence of the training treebanks was pseudo-projectivised before training. Using the weighted LAS, we then chose the best combination of parameters for each language. Since the python version of CNN (used by our adaptation of BistParser) does not support GPU, training was slow[12]. Thus we stopped training usually after 15 epochs unless the intermediary results were promising enough to continue. Figure 1 shows the system architecture. The upper part represents the data flow for the training, the lower part represents the predicting phase.

We did all training on two Ubuntu 16.04 servers[13] with 64 GB RAM. As said above, the version of the CNN library we used, does not run on GPU, so all training was single threaded. The training processes used up to 15 GB RAM, and took between 1 minute (Kazakh) and 53 hours (Czech). depending on the size of the treebank. This corresponds to 0.5 to 3 seconds per sentence during training. Training for the surprise languages (using treebanks of typologically close languages, cf. section, 4), took significantly longer (up to 90 hours for Czech).

Training was on the gold values (form, lemma, XPOS, UPOS, deprel, head) of the training treebanks[14], however, both, the development set (on the Tira-platform) and the final test set use the UDPipe output e.g. lemma, XPOS or UPOS (Straka et al., 2016) which may be erroneous. So we expected a certain drop of LAS for the tests. In order to be prepared, we tried to add erroneous lemmas and UPOS in the training data. This, however, did not produce better results, so we abandoned the

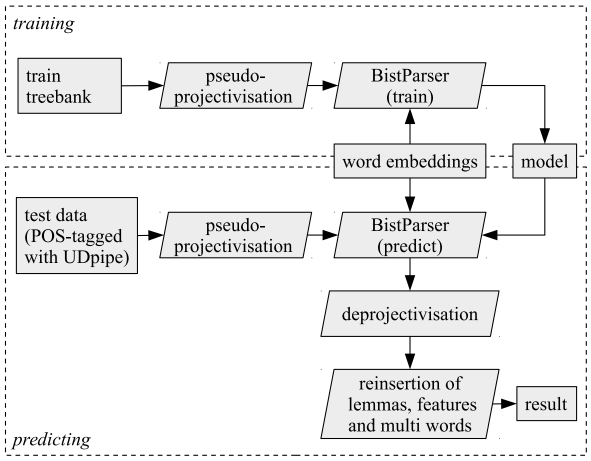

Figure 1: Schema of the system architecture

idea. Knowing that training takes a certain time we did not POS-tag the training treebanks with UDpipe to have similar "noise" than the test treebanks. The final results obtained with the development corpora (or split from train corpora when there were no development corpora) are shown in table 1. We did not (yet) use the morphological features (column 6). First tests on French showed that a slight increase in LAS is possible, so we will work on this in the future.

With 16GB RAM on the virtual machine provided by Tira (Potthast et al., 2014)[15] the 56 development corpora (on the Tira platform) were processed in about 130 minutes.

4 Surprise languages

The biggest challenge were the 4 surprise languages. Having only between 20 and 109 sentences to train on (even less if we wanted to split it into a train and development corpus) did not help (see table 2 for some details). Since the word embedding files where also rather small we chose not to train on the languages themselves, but to keep all of the provided sentences for the development corpus. So we first tried three similar approaches in order to be able to predict dependency relations for these languages:

1. lumping together 2000 sentences of 11 typologically different languages (German, Irish, Russian, Spanish, Turkisch, Arabic, Persian, Indonesian, Basque, Finnish and Estonian)

2. lumping together 5000 sentences of 11 typologically different languages (as above)

[12]The successor of CNN, Dynet, supports GPU, but since BistParser learns on a phrase by phrase base, no gain in time can be observed.

[13]Intel Xeon CPU E5-1620 v4 at 3.50GHz and Intel Core i7-6900K CPU at 3.20GHz respectively.

[14]Apart from numerous punctuation symbols with wrong heads, we found several bad annotations for words as well in different languages.

[15]The tests are run on the Tira platform (`http://www.tira.io`), we used Ubuntu 16.04 as operating system

treebank	*number of dimensions of word embeddings*	*size of hidden layer*	*weighted LAS*	*treebank*	*number of dimensions of word embeddings*	*size of hidden layer*	*weighted LAS*
ar	300	50	76.75	hu	300	40	64.72
bg	300	100	83.62	id	300	50	71.82
ca	300	50	85.67	it	500	100	86.34
cs	300	50	87.08	it-ParTUT	300	100	79.54
cs-CAC	300	100	85.93	ja	500	100	91.04
cs-CLTT	500	100	78.95	kk	300	100	31.53
cu	500	40	75.18	ko	300	50	73.76
da	500	40	72.46	la	300	50	54.40
de	300	50	78.03	la-ITTB	300	50	72.32
el	300	100	80.59	la-PROIEL	300	50	70.49
en	300	50	84.60	lv	300	50	69.47
en-LinES	500	40	77.83	nl	500	100	78.70
en-ParTUT	300	50	79.93	nl-LassySmall	500	100	73.99
es	300	50	80.29	no-Bokmaal	300	50	82.19
es-AnCora	500	100	85.60	no-Nynorsk	300	50	80.25
et	300	50	67.68	pl	500	40	86.78
eu	300	50	65.65	pt	300	50	91.21
fa	500	40	84.23	pt-BR	500	100	87.22
fi	300	100	77.84	ro	300	50	80.10
fi-FTP	300	50	82.70	ru	300	50	80.22
fr	500	100	83.64	ru-SynTagRus	300	50	82.06
fr-ParTut	500	100	83.03	sk	500	40	82.67
fr-Sequoia	300	50	80.42	sl	300	50	86.62
ga	500	100	64.84	sl-sst	300	100	60.24
gl	300	50	78.46	sv	500	100	80.01
gl-TreeGal	500	100	70.04	sv-LinES	300	50	79.73
got	none	100	68.53	tr	300	50	56.60
grc	500	40	58.21	ug	500	40	42.35
grc-PROIEL	300	50	66.81	uk	300	50	75.77
he	300	50	78.87	ur	500	100	80.89
hi	500	40	89.70	vi	500	100	71.06
hr	300	100	72.56	zh	500	100	78.28

Table 1: Development results (without surprise languages)

3. lumping together 2000 sentences of 23 typologically different languages (as above and Chinese, Japanese, Korean, Vietnamese, Romanian, Latin, Greek, Ancient Greek, Hebrew, Urdu and Hungarian)

	hsb	*sme*	*kmr*	*bxr*
sentences	109	20	65	20
words	460	147	242	153
has XPOS	no	yes	yes	no
has features	yes	yes	yes	yes

Table 2: Statistics on the data of the surprise languages

In all three cases we replaced the forms of all closed word classes (i.e. all but nouns, adjectives and verbs) with the corresponding UPOS in the training and in the test corpus (for the *CoNLL 2017 UD Shared Task* we inserted the original forms again after predicting the dependency relations. The "mix" is then trained with a hidden layer size of either 100 or 50, but without word embeddings. We initially tested these models using the test corpus for the Tamil treebank (UD v2.0). Using the "mix" with 23 languages (3) resulted in the best weighted LAS, 35.2% (35.3% if using a hidden layer size of 50). The weighted LAS for the surprise languages is shown in table 3.

language	*hidden layer size*	
	50	*100*
Upper Sorbian (hsb)	59.5%	63.2%
Northern Sami (sme)	49.2%	47.5%
Kurmanji (kmr)	28.9%	29.2%
Buryat (bxr)	25.3%	26.3%

Table 3: Weighted LAS of the surprise languages using a model trained on 23 languages

By replacing words of the closed word classes by their UPOS we tried to get similar corpora for training (on languages other than the surprise language) and predicting (surprise languages), assuming that the syntactical structures are similar enough, especially if we use only typologically close languages (see below). This technique avoids also the problem of different alphabets for typologically close languages, since we use UPOS and not character chains.

Eventhough these results were encouraging, we hoped an increase of the weighted LAS should still be feasible. Especially since some of the surprise

Upper Sorbian		Northern Sami	
cs (100)	**69.5%**	**fi (100)**	**52.9%**
cs (50)	67.5%	fiu[16](100)	51.7%
pl (50)	56.9%	fiu (50)	49.7%
pl (100)	51.9%	fi (50)	50.8%

Table 4: Weighted LAS using typologically close languages (with hidden layer size)

languages are typologically (very) close to languages within the Universal Dependency corpus: Upper Sorbian is a slavonic language very close to Czech (and slightly less close to Polish). Northern Sami shares quite a lot of typological features with the Finnic branch of the Fenno-Ugric languages (here Finnish and Estonian), and Kurmanji shares at least some typolological feature with Persian (both are from the Iranian subgroup of the Indo-European language family. However Buryat, a Mongolian language, is not typologically close to any of the shared task's languages. Even though Turkish seems close enough, to our surprise Hindi was finally the best guess. With Urdu, which is very similar to Hindi apart from the fact that it uses the Arabic alphabet instead of Devanagari, the LAS was less good.

As for the language mix, we replaced the forms of the closed word classes in the training corpora by the corresponding UPOS (except nouns, verbs and adjectives) and trained the modified treebanks (cf. tables 4 and 5, best configuration in bold).

Kurmanji		Buryat	
fa (100)	**36.7%**	**hi (50)**	**32.0%**
fa (50)	35.8%	hi (100)	28.1%
hi (50)	22.2%	ur (50)	28.0%
hi (100)	22.0%	tr (100)	27.6%
ur (100)	20.6%	fi (100)	21.8%
ur (50)	20.6%	tr (50)	19.4%
		ja (50)	18.0%

Table 5: Weighted LAS using typologically close languages (with hidden layer size)

The reason for not replacing nouns, adjectives, and verbs is simple: Leaving the original words of the training corpus language and the test corpus language, means while the parser predicts, it comes across a word which it has never seen during training. But since it has the UPOS, it has

[16]*fiu* represents the Fenno-Ugric languages, in our case a mix of Finnish, Estonian and Hungaric

treebank	*development results*	*final test results*	*difference*
ar	76.75	67.26	-9.49
ar-pud (*unavailable for dev.*)		44.77	
bg	83.62	85.06	1.44
bxr (*surprise lg*)	32.00	25.25	-6.75
ca	85.67	86.24	0.57
cs	87.08	84.33	-2.75
cs-CAC	85.93	83.98	-1.95
cs-CLTT	78.95	72.99	-5.96
cs-pud (*unavailable for dev.*)		79.49	
cu	75.18	64.26	-10.92
da	72.46	73.54	1.08
de	78.03	73.38	-4.65
de-pud (*unavailable for dev.*)		69.75	
el	80.59	80.69	0.10
en	84.60	77.51	-7.09
en-LinES	77.83	73.36	-4.47
en-ParTUT	79.93	75.78	-4.15
en-pud (*unavailable for dev.*)		79.67	
es	80.29	83.03	2.74
es-AnCora	85.60	85.57	-0.03
es-pud (*unavailable for dev.*)		78.78	
et	67.68	58.98	-8.70
eu	65.65	65.29	-0.36
fa	84.23	80.87	-3.36
fi	77.84	73.97	-3.87
fi-FTP	82.70	78.64	-4.06
fi-pud (*unavailable for dev.*)		77.52	
fr	83.64	80.58	-3.06
fr-ParTut (*)	83.03	77.26	(tested with fr)
fr-pud (*unavailable for dev.*)		74.63	
fr-Sequoia	80.42	81.54	1.12
ga	64.84	63.10	-1.74
gl	78.46	79.66	1.20
gl-TreeGal (*)	70.04	22.46	(tested with gl)
got	68.53	57.97	-10.56
grc	58.21	54.10	-4.11
grc-PROIEL	66.81	65.50	-1.31
he	78.87	58.07	-20.8
hi	89.70	87.09	-2.61
hi-pud (*unavailable for dev.*)		51.02	
hr	72.56	77.11	4.55
hsb (*surprise lg*)	69.50	58.25	-11.25
hu	64.72	64.59	-0.13
id	71.82	73.64	1.82
it	86.34	86.65	0.31
it-ParTuT	79.54	(*withdrawn*)	
it-pud (*unavailable for dev.*)		84.89	
ja	91.04	73.37	-17.67
ja-pud (*unavailable for dev.*)		76.74	
kk	31.53	21.31	-10.22
kmr (*surprise lg*)	36.70	38.31	1.61
ko	73.76	67.76	-6.00
la	54.40	43.16	-11.27
la-ITTB	72.32	76.42	4.10
la-PROIEL	70.49	60.44	-10.05
lv	69.47	61.52	-7.95
nl	78.70	70.33	-8.37
nl-LassySmall	73.99	77.58	3.59
no-Bokmaal	82.19	83.79	1.60
no-Nynorsk	80.25	81.69	1.44
pl	86.78	81.71	-5.07
pt	91.21	76.40	-14.81
pt-BR	87.22	87.07	-0.15
pt-pud (*unavailable for dev.*)		69.00	
ro	80.10	81.34	1.24
ru	80.22	76.28	-3.94
ru-pud (*unavailable for dev.*)		69.58	
ru-SynTagRus	82.06	87.10	5.04
sk	82.67	75.97	-6.7
sl	86.62	82.38	-4.24
sl-sst (*)	60.24	40.25	(tested with sl)
sme (*surprise lg*)	52.90	33.08	-19.82
sv	80.01	78.85	-1.16
sv-LinES	79.73	74.28	-5.45
sv-pud (*unavailable for dev.*)		70.82	
tr	56.60	55.21	-1.39
tr-pud (*unavailable for dev.*)		34.36	
ug	42.35	34.24	-8.11
uk	75.77	62.30	-13.47
ur	80.89	77.93	-2.96
vi	71.06	39.12	-31.94
zh	78.28	59.33	-18.95
average	74.40	68.61	

Table 6: comparison of training results (on development corpora) and final results. Due to an error, the models for the treebanks marked (*) were unfortunately not used for the tests.

an idea what to do. If we took UPOS for verbs, nouns, and adjectives, this meant that the whole training (and test) corpus would only contain one single verb (namely VERB), noun (NOUN) and adjective (ADJ). Tests showed, that the parser got confused, since it encountered the same verb, noun or adjective in very different syntactic contexts. Of course, this applies also to prepositions or adverbs but keeping the open word classes gave the best development results.

As expected, Upper Sorbian and Norther Sami give quite acceptable results using models trained on Czech and Finnish respectively. Due to the fact that the provided treebanks for Kazakh and Uyghur are both very small we tried to apply the same approach of using the training corpus of a typologically close language (here Turkish). However, the results were disappointing. Thus, we continue to use the models trained on very small corpora for these two languages in the shared task. Possibly the fact that the raw text corpus used to calculate word embeddings for Kazakh and Uyghur are much bigger than those of the surprise languages allowed to produce usable word embeddings. If so, this would mean that word embeddings play a very prominent role in data driven dependency parsing.

5 Results of the Shared Task

Our final macro-averaged LAS F1 score on the *CoNLL 2017 UD Shared Task* test data (Nivre et al., 2017a) was 68.61%, (10th out of 33)[17]. The details show that our approach worked well for the bigger treebanks and the surprise languages (where we ended up as 8th). In general, the results per language are slightly lower than those we had during training on the development corpora (cf. table 1). This is due to the fact we did our training on forms, lemmas, UPOS and XPOS of the training corpus, which are gold. In the test data, lemmas, UPOS and XPOS (if present), however, are predicted by UDpipe, and do contain some errors with respect to the gold standard.

After the end of the test phase, we discovered a bug in our chain, which concerned languages, which have only UPOS data. In this case the UPOS information was totally discarded by error. Thus all training and testing are done only on the forms[18].

Further we made en error uploading the models for the gl_TreeGal, fr_parTut and sl_sst treebanks. During the tests the models trained on the basic gl, fr and sl treebanks were used instead. After the test phase we corrected these errors. Fortunately, their impact was not that hard. Apart from the result for gl_TreeGal and sl_sst, which went up to 66.13% (from 22.46%) and to 47.68 (from 40.25) respectively once the correct model was used, the results for the other corpora changed only slightly, the global results could have been 69.38%. All results are shown in table 6. The column on the right shows the difference between the results of the development corpora and test data. For some languages, the test results are unexpectedly lower than the results on the development corpora. For gl_TreeGal, fr_parTut and sl_sst, this is due to errors when installing our system on the Tira-platform. The lower performance on languages like Chinese, Ukrainian, Vietnamese or Latin (both ITTB and PROIEL) seems to be caused by the nature of the test corpora themselves. Systems of other participants seem to drop in performance as well; for all these languages our system is still around the 10th position of the global ranking. Perhaps a cause may be the fact that the XPOS we use (predicted by UDpipe) contain more errors than average for the Chinese, Ukrainian or Vietnamese treebanks than for languages where our test score is closer to the development score.

Acknowledgments

We would like to thank the developpers of BistParser, Eliyahu Kiperwasser and Yoav Goldberg, and the developper of the CNN library (Chris Dyer) for making them available as open source on GitHub. Finally we would like to thank our colleague Ghislain Putois for help on all aspects on neural networks.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606. http://arxiv.org/abs/1607.04606.

[17] http://universaldependencies.org/conll17/results.html

[18] This concerns the treebanks da, en-LinES, es, eu, fr, fr-Sequoia, hr, hu, id, ja, nl-LassySmall, no-Bokmaal, no-Nynorsk, ru-SynTagRus and sv-LinEs.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric de la Clergerie. 2014. Deep syntax annotation of the sequoia french treebank. In *Proc. of LREC 2014*. Reykjavík, Iceland.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR* abs/1607.01759. http://arxiv.org/abs/1607.01759.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree lstms. *TACL* 4:445–461. https://transacl.org/ojs/index.php/tacl/article/view/798.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool Publishers.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 673–682. http://dl.acm.org/citation.cfm?id=2002472.2002558.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu, and Castelló Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL 2013*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Joakim Nivre. 2008. Algoritms for deterministic incremental dependency parsing. *Computational Linguistics* 34(4):513–553.

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, Portorož, Slovenia, pages 1659–1666.

Joakim Nivre and Chiao-Ting Fang. 2017. Universal dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. Association for Computational Linguistics, Gothenburg, Sweden, pages 86–95. http://www.aclweb.org/anthology/W17-0411.

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.

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.

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, Portorož, Slovenia.

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, Héctor 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.