The HIT-SCIR System for End-to-End Parsing of Universal Dependencies

Wanxiang Che, Jiang Guo, Yuxuan Wang, Bo Zheng
Huaipeng Zhao, Yang Liu, Dechuan Teng and **Ting Liu**
Center for Social Computing and Information Retrieval
Harbin Institute of Technology, Harbin, China 150001
{wxche,jguo,yxwang,bzheng,hpzhao,yliu,dcteng,tliu}@ir.hit.edu.cn

Abstract

This paper describes our system (HIT-SCIR) for the CoNLL 2017 shared task: Multilingual Parsing from Raw Text to Universal Dependencies. Our system includes three pipelined components: *tokenization*, *Part-of-Speech* (POS) *tagging* and *dependency parsing*. We use character-based bidirectional long short-term memory (LSTM) networks for both tokenization and POS tagging. Afterwards, we employ a list-based transition-based algorithm for general non-projective parsing and present an improved Stack-LSTM-based architecture for representing each transition state and making predictions.

Furthermore, to parse low/zero-resource languages and cross-domain data, we use a model transfer approach to make effective use of existing resources. We demonstrate substantial gains against the UDPipe baseline, with an average improvement of 3.76% in LAS of all languages. And finally, we rank the 4th place on the official test sets.

1 Introduction

Our system for the CoNLL 2017 shared task (Zeman et al., 2017) is a pipeline which includes three cascaded modules, *tokenization*, *Part-of-Speech* (POS) *tagging* and *dependency parsing*.

- Tokenization. This module includes two components, the *sentence segmenter* and the *word segmenter* which recognize the sentence and word boundaries respectively (Section 2.1).
- POS tagging. We focus mainly on universal POS tags, and don't use language-specific POS as well as other morphological features (Section 2.2).
- Dependency parsing. To handle the non-projective dependencies in most of the languages (or treebanks) provided in the task, we employ the list-based transition parsing algorithm (Choi and McCallum, 2013), equipped with an improved Stack-LSTM-based model for representing the transition states, i.e., configurations (Section 2.3).

We mainly concentrate on parsing in this task, and make use of UDPipe (v1.1) (Straka et al., 2016a) for most of the pre-processing steps. However, our preliminary experiments showed that the UDPipe tokenizer and POS tagger perform rather poorly in some languages and specific domains. Therefore, we develop our own tokenizer and POS tagger for a subset of languages.

To deal with the *parallel test sets* (cross-domain) and low/zero-resource languages, we adopt the neural transfer approaches proposed in our previous studies (Guo et al., 2015, 2016) to encourage knowledge transfer across different but related languages or treebanks.

Experiments on 81 test sets demonstrate that our system (HIT-SCIR: software4) obtains an average improvement of 3.76% in LAS as compared with the UDPipe baseline, and ranks the 4th place in this task.

2 System Architecture

2.1 Tokenization

2.1.1 Sentence Segmentation

We develop our own sentence segmentation models for the languages which have white spaces as token separators and on which UDPipe doesn't

Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 52–62,
Vancouver, Canada, August 3-4, 2017.

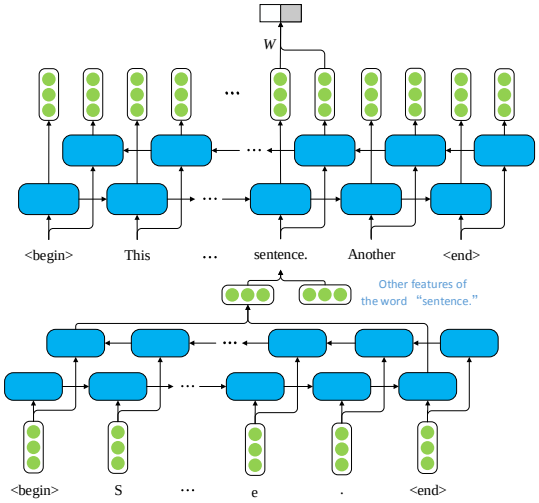

Figure 1: The hierarchical Bi-LSTM model for sentence segmentation.

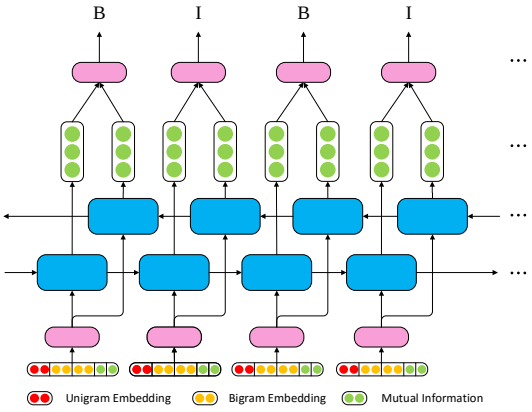

Figure 2: The word segmentation model. 'B' denotes the beginning position and 'I' denotes the middle or ending positions of a word.

perform well. We formalize the sentence segmentation process as a binary classification problem, that is to classify each token as either the end of a sentence or not. We notice that character-level information is critical for sentence segmentation, since texts are not tokenized yet in the current phase. Therefore, we develop a hierarchical LSTM-based model, as illustrated in Figure 1, in which characters in each token are composed using a character-based bidirectional LSTM (Bi-LSTM) network and then concatenated with additional token-level features (e.g., token embedding, the first character of this token, etc.) and passed through a token-level Bi-LSTM. The hidden states of the token-level Bi-LSTM are finally used for classification through a softmax layer.

We follow the strategy of the UDPipe tokenizer (Straka et al., 2016a) and employ a sliding window to incrementally segment a document into sentences.

In addition, we notice that for certain treebanks (e.g., la_ittb and cs_cltt), some punctuation-related rules derived from the training data can be highly effective. To be more specific, some punctuations that appear as the end of a sentence with high probability will be used directly for determining sentence boundaries. Therefore, we develop additional rule-based systems for these data instead of using the neural models as describe above.

2.1.2 Word Segmentation

We develop our own word segmentation models particularly for languages which do not have explicit word boundary markers, i.e., white spaces, including *Chinese, Japanese* and *Vietnamese*.[1]

Our word segmentation model is also built on Bi-LSTM networks, and incorporates rich statistics-based features gathered from large-scale unlabeled data. Specifically, we utilize features like character-unigram embeddings, character-bigram embeddings and the pointwise mutual information (Liang, 2005) (PMI) of adjacent characters. Formally, the input of our model at each time step t can be computed as:

$$z_t = [U_t; B_{t-1}; B_t; \texttt{PMI}(c_{t-1}, c_t); \texttt{PMI}(c_t, c_{t+1})] \quad (1)$$

$$x_t = max\{0, W z_t + b\} \quad (2)$$

where U_t and B_t denote the unigram embedding and bigram embedding respectively at position t and PMI denotes the pointwise mutual information between two characters.

The PMI values are computed through:

$$\texttt{PMI}(c_1, c_2) = \log \frac{p(c_1 c_2)}{p(c_1)\, p(c_2)} \quad (3)$$

where c_1 and c_2 are two characters, $p(c_1)$, $p(c_2)$ and $p(c_1 c_2)$ are counted on the raw data provided by the shared task. $p(s)$ denotes the probability string s appears in the raw data. We scale PMI

[1] *Vietnamese* requires word segmentation because white spaces occur both inter- and intra-words. When segmenting *Vietnamese*, white space-separated tokens are used as inputs, rather than characters as in *Chinese* and *Japanese*. In addition, we don't consider *Korean* here since the *Korean* input texts have already been segmented in the corpus provided by the task.

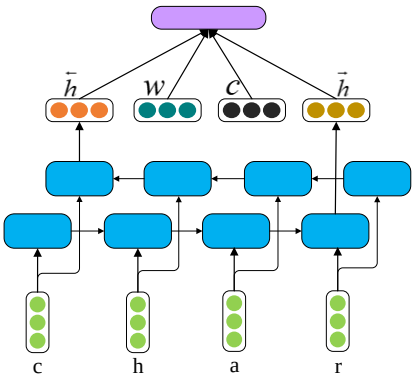

Figure 3: Structure of the character-based composition model for learning word representations. 'w' denotes the pre-trained word embedding, 'c' denotes the Brown cluster embedding.

with their Z-scores, the Z-score of a PMI value x is $\frac{x-\mu}{\sigma}$, where μ and σ are the mean and standard deviation of the PMI distribution, respectively.

Figure 2 shows the architecture of our word segmentation model.

The character-unigram embeddings and character-bigram embeddings are obtained using word2vec (Mikolov et al., 2013) on the raw data.

2.2 Part-of-Speech Tagging

The UDPipe POS tagger is trained using averaged perceptron with feature engineering. In our system, we use a model similar to the one for sentence segmentation (Section 2.1.1), i.e., a hierarchical Bi-LSTM model which outperforms UDPipe on most of datasets with much fewer features. Concretely, each word is modeled using a character-based Bi-LSTM, so that word prefix and suffix features can be effectively incorporated, which is particularly important for morphologically rich languages. In addition, modeling from characters alleviates the problem of Out-of-Vocabulary (OOV) words.

The character-based compositional embedding of each word is then concatenated with a pre-trained word embedding and a Brown cluster embedding, resulting in the final word representation which is fed as input of a word-level Bi-LSTM for POS tagging. Formally,

$$x = \max\{0, W[\overrightarrow{h}; \overleftarrow{h}; w; c] + q\} \quad (4)$$

Figure 3 illustrates the structure of the character-based composition model.

2.3 Dependency Parsing

The transition-based dependency parsing algorithm with a list-based arc-eager transition system proposed by Choi and McCallum (2013) is used in our parser. We base our parser mainly on the Stack-LSTM model proposed by Dyer et al. (2015), where three Stack-LSTMs are utilized to incrementally obtain the representations of the buffer β, the stack σ and the transition action sequence A. In addition, a dependency-based Recursive Neural Network (RecNN) is used to compute the partially constructed tree representation. However, compared with the arc-standard algorithm (Nivre, 2004) used by Dyer et al. (2015), the list-based arc-eager transition system has an extra component in each configuration, i.e., the deque δ. So we use an additional Stack-LSTM to learn the representation of δ. More importantly, we introduce two LSTM-based techniques, namely *Bi-LSTM Subtraction* and *Incremental Tree-LSTM* (explained below) for modeling the buffer and sub-tree representations in our model.

The pre-trained word embedding (100-dimensional), Brown cluster embedding (100-dimensional), along with a 100-dimensional randomly initialized word embedding updated while training,[2] and a 50-dimensional embedding for UPOS are concatenated and passed through a non-linear layer to obtain the representation of each word.

Representations of the four components in our transition system are concatenated and passed through a hidden layer to obtain the representation of the parsing state at time t:

$$e_t = \max\{0, W[s_t; b_t; p_t; a_t] + d\} \quad (5)$$

where s_t, b_t, p_t and a_t are the representation of σ, β, δ and A respectively. d is the bias. e_t is finally used to compute the probability distribution of possible transition actions at time t through a softmax layer. Figure 4 shows the architecture.

2.3.1 Bi-LSTM Subtraction

We regard the buffer as a *segment* and use the subtraction between LSTM hidden vectors of the segment head and tail as its representation. To include the information of words out of the buffer, we apply subtraction on bidirectional LSTM representations over the whole sentence (Wang et al.,

[2] Unfortunately we did not have access to enough raw text of *Gothic*, thus no pre-trained word embedding nor Brown cluster is utilized for it.

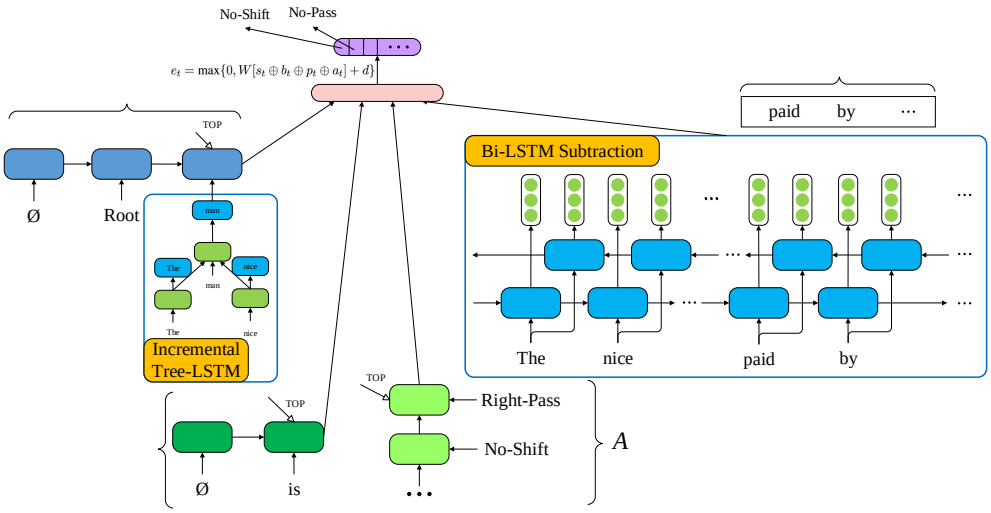

Figure 4: Example transition state representation based on LSTMs. The buffer β is represented by Bi-LSTM Subtraction, the sub-trees are computed by Incremental Tree-LSTM.

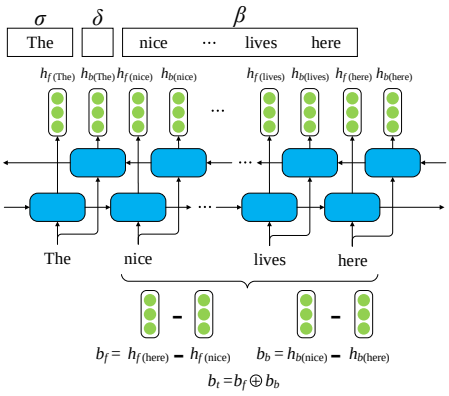

Figure 5: Illustration of Bi-LSTM Subtraction for buffer representation learning. $h_f(*)$ and $h_b(*)$ indicate the hidden vectors of forward and backward LSTM respectively. b_t is the resulting buffer representation.

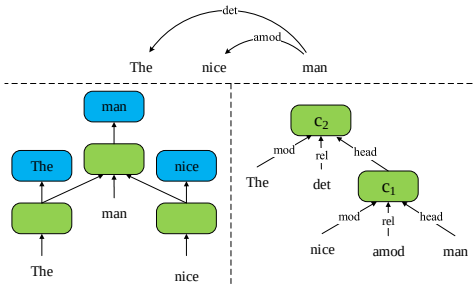

Figure 6: Representations of a dependency subtree (above) computed by Tree-LSTM (left) and dependency-based RecNN (right).

2016; Kiperwasser and Goldberg, 2016; Cross and Huang, 2016), thus called **Bi-LSTM Subtraction**.

The forward and backward subtractions are calculated independently, i.e., $b_f = h_f(l) - h_f(f)$ and $b_b = b_b(f) - b_b(l)$, where $h_f(f)$ and $h_f(l)$ are the hidden vectors of the first and the last words in the forward LSTM, $h_b(f)$ and $h_b(l)$ are the hidden vectors of the first and the last words in the backward LSTM. Then b_f and b_b are concatenated as the buffer representation. As illustrated in Figure 5, the forward and backward subtractions for the buffer are $b_f = h_f(here) - h_f(nice)$ and $b_b = h_b(nice) - h_b(here)$ respectively.

2.3.2 Incremental Tree-LSTM

We use a Tree-LSTM (Tai et al., 2015; Zhu et al., 2015) in our parser to model the sub-trees during parsing. The example in Figure 6 shows the differences between RecNN (Dyer et al., 2015) and Tree-LSTM. In RecNN, the representation of a sub-tree is computed by recursively combining head-modifier pairs. Whereas in Tree-LSTM, a head is combined with all of its modifiers simultaneously in each LSTM unit.

However, our implementation of Tree-LSTM is different from the conventional one. Unlike traditional bottom-up Tree-LSTMs in which each head and all of its modifiers are combined simultaneously, the modifiers are found incrementally during our parsing procedure. Therefore, we propose **Incremental Tree-LSTM**, which obtains sub-tree representations incrementally. To be more specific, each time a dependency arc is generated,

we collect representations of all the found modifiers of the head and combine them along with the embedding of the head as the representation of the sub-tree. The original embedding rather than the current representation of the head is utilized to avoid the reuse of modifier information, since the current representation of the head contains information of its modifiers found previously.

2.3.3 Parser Ensembling

For a majority of languages, we found that the parsing performance can be improved by simply integrating two separately trained models. More specifically, for each language two models with different random seeds are trained separately. While predicting, in each state, both models are used to calculate the scores for valid transitions under this configuration as described above. Then the score distributions computed by two models are summed to get the final scores for the valid transitions, among which the one with the highest score will be taken as the next transition.

3 Transfer Parsing across Domains and Languages

3.1 Cross-Domain Transfer

For 15 out of 45 languages presented in the task, multiple treebanks from different domains are provided. To exploit the benefits from these cross-domain data, we use a simple inductive transfer approach which has two stages:

1. Multiple treebanks of each language are combined to train an unified parser.

2. The unified parser is then fine-tuned on the training treebank of each domain, to obtain target domain-specific parsers.

In practice, for each language considered here, we treat the largest treebank as our source-domain data, and the rest as target-domain data. Only target-domain models are fine-tuned from the unified parser, while the source-domain parser is trained separately using the source treebank alone.

For the new *parallel test sets* in test phase, we simply use the model trained on source-domain data, without any assumption on the target domain.

3.2 Cross-Lingual Transfer

We consider the languages which have less than 900 sentences in the training treebank as *low-resource*, and employ the cross-lingual model transfer approach described in Guo et al. (2015, 2016) to benefit from existing resource-rich languages.

Target	hu	uk	qa	ug	kk
Source	fi_ftb	ru_syntagrus	en	tr	tr

Table 1: Cross-lingual transfer settings for low-resource target languages.

The low-resource languages here include *Ukrainian* (uk), *Irish* (ga), *Uyghur* (ug) and *Kazakh* (kk). We determine their source language (treebank) according to the language families they belong to and their linguistic typological similarity. Specifically, the transfer setting is shown in Table 1.

The transfer approach is similar to cross-domain transfer as described above, with one important difference. Here, we use cross-lingual word embeddings and Brown clusters derived by the *robust projection* approach (Guo et al., 2015) when training the unified parser, to encourage knowledge transfer across languages at lexical level. Specifically, for each source and target language pair ⟨src, tgt⟩, we derive an alignment matrix $A_{tgt|src}$ from a collected bilingual parallel corpus, where each element $A_{tgt|src}(i,j)$ is the normalized count of alignments between corresponding words in their vocabularies:

$$A_{tgt|src}(i,j) = \frac{\#(V_{tgt}^{(i)} \leftrightarrow V_{src}^{(j)})}{\sum_k \#(V_{tgt}^{(i)} \leftrightarrow V_{src}^{(k)})} \quad (6)$$

Given a pre-trained source language word embedding matrix E_{src}, the resulting word embedding matrix for the target language can be simply computed as:

$$E_{tgt} = A_{tgt|src} \cdot E_{src} \quad (7)$$

Therefore, the embedding of each word in the target language is the weighted average of the embeddings of its translation words in our bilingual parallel corpus.

The cross-lingual Brown clusters are obtained using the PROJECTED clustering approach described in (Täckström et al., 2012), which assigns a target word to the cluster with which it is most often aligned:

$$c(w_i^{tgt}) = \arg\max_k \sum_j A_{tgt|src}(i,j) \cdot \mathbb{1}[c(w_j^{src}) = k] \quad (8)$$

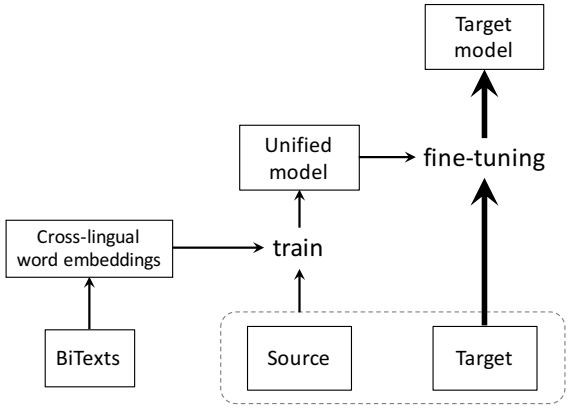

Figure 7: The cross-lingual transfer approach.

Target	bxr	kmr	sme	hsb
Source	tr & ug & kk	fa	fi_ftb & fi	cs

Table 2: Cross-lingual delexicalized transfer settings for surprise languages.

After that, target language-specific parsers are obtained through fine-tuning on their own treebanks. Figure 7 illustrates the flow of our transfer approach.

For the *surprise languages* in the final test phase, we use the transfer settings in Table 2. We use multi-source **delexicalized transfer** for surprise language parsing, considering that bilingual parallel data which is required for obtaining cross-lingual word embeddings is not available for these languages.

4 Experiments

We first describe our experiment setups and strategies for processing different languages (treebanks) in each module. Then we present the results and analysis.

4.1 Experimental Settings

4.1.1 Model Selection Strategies

For *sentence segmentation*, we apply our own models for a subset of languages on which UDPipe yields poor performance, and use UDPipe for the rest languages.[3] Specifically, we use the rule-based model for la_ittb and cs_cltt,[4] and use the Bi-LSTM-based model (Figure 1) for sk, en, en_lines, fi_ftb, got, nl_lassysmall, grc_proiel, la_ittb, cu, la_proiel, da and sl_sst. For *word segmentation*, we use our Bi-LSTM-based model for zh, ja, ja_pud and vi, which don't have explicit word boundary markers, i.e., white spaces.

We use our own POS taggers for all of the languages, except for the surprise languages, which we rely on UDPipe for all pre-processing steps.

Our strategies for parsing are shown in Table 3. We determine the optimal parser (single, ensemble or transfer) for each treebank according to the performance on the development data.

4.1.2 Data and Tools

We use the provided 100-dimensional multilingual word embeddings[5] in our tokenization, POS tagging and parsing models, and use the Wikipedia and CommonCrawl data for training Brown clusters. The number of clusters is set to 256.

For cross-lingual transfer parsing of low-resource languages, we use parallel data from OPUS to derive cross-lingual word embeddings.[6] The *fast_align* toolkit (Dyer et al., 2013) is used for word alignment.[7]

We use the Dynet toolkit for the implementation of all our neural models.[8]

4.2 Effects of Different Parts in Dependency Parsing

We conduct experiments on the development sets of 4 treebanks to investigate the contributions of the two architectures we proposed (i.e., the Incremental Tree-LSTM and the Bi-LSTM Subtraction) and the Brown cluster. The LAS of different experiment settings are presented in Table 4. Results show that Brown clusters and both architectures help to improve the parsing performance in most situations. And the ensemble method we eventually choosed which incorporated the two architectures as well as Brown clusters and utilized two models for predicting yield the best performance.

4.3 Effect of Transfer Parsing

To investigate the effect of transfer parsing on cross-domain and cross-lingual data, we compare our transferred system with the supervised system on a subset of treebanks. Evaluation is conducted

[3] We use the same hyper-parameter settings as provided by the organizers to train the UDPipe models.

[4] However, the rule-based model does not yield good performance on the two test sets. We suggest that the rules we use are overfitting the development sets to some degree.

[5] lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989

[6] opus.lingfil.uu.se

[7] https://github.com/clab/fast_align

[8] github.com/clab/dynet

Strategy	ltcode
Single	cs_cac, bg, ja, he
Ensemble (2)	cs, ru_syntagrus, la_ittb, fi_ftb, grc_proiel, es_ancora, es, de, hi, ca, en, fi, sk, ro, hr, pl, ar, eu, fa, id, ko, da, sv, cu, ur, zh, tr, got, sv_lines, lv, gl, et, el, vi, hu
Cross-domain Transfer	no_bokmaal, no_nynorsk, la, la_proiel, grc, pt, pt_br, sl, sl_sst, nl, nl_lassysmall, en_lines, en_partut, fr, fr_sequoia, fr_partut, it, it_partut†, gl_treegal, cs_cltt, ru
Cross-lingual Transfer	uk, ga, ug, kk
Delexicalized Transfer	bxr, kmr, sme, hsb

Table 3: Model selection strategies for all treebanks. † it_partut is excluded from the final test sets. But it's used in our transfer parsing as a source treebank.

Settings	cs	de	en	ko
Baseline	86.79	80.08	81.87	67.21
B	87.51	80.01	82.48	68.19
T	87.43	80.01	82.24	68.29
B + T	87.67	80.24	82.73	68.34
B + T + C	87.62	81.45	83.37	70.24
Ensemble (2)	**88.52**	**81.92**	**83.99**	**71.38**

Table 4: Experiment results (LAS) on development sets with different settings. B: Bi-LSTM Subtraction, T: Incremental Tree-LSTM, C: Brown cluster. Ensemble is produced with models we eventually submitted.

on the development data or through 5-fold cross-validation when development data is not available. Results are shown in Table 5 and 6 respectively. We can see that both cross-domain and cross-lingual transfer parsing improve over the supervised systems significantly.

4.4 Results

The overall results of our end-to-end universal parsing system on 81 test treebanks are shown in Table 7, with comparison to the UDPipe baseline models. We obtain substantial gains over UDPipe on 76 out of 81 treebanks, with 3.76% improvements in average LAS. It spent about 9 hours to evaluate all of 81 test sets end-to-end and needed up to 4GB memory on the TIRA virtual machine.

4.5 Post-Evaluation

We realized a small problem in our implementation of the word segmentation models after official evaluation. After revision, we re-evaluated our models on the four test treebanks: zh, vi, ja and ja_pud. The post-evaluation results are shown in Table 8. On zh, vi and ja_pud, we outperform the rank-1 system significantly. We can see that the performance of word segmentation is crucial for the pipeline system.

ltcode	Supervised		Transfer	
	UAS	LAS	UAS	LAS
cs_cltt	81.23	77.80	85.38	83.08
en_lines	82.30	78.44	83.65	79.75
en_partut	82.66	78.72	85.97	82.08
fr_sequoia	88.68	86.67	89.46	87.79
la_proiel	78.77	73.18	79.67	74.46
no_bokmaal	89.76	87.50	90.49	88.37
no_nynorsk	88.39	85.84	89.41	87.08
nl_lassysmall	86.11	82.65	87.39	84.02
pt_br	91.59	89.65	91.93	90.16
Average	*85.50*	*82.27*	***87.04***	***84.09***

Table 5: Effects of cross-domain transfer parsing on a subset of development sets.

5 Conclusion and Future Work

Our CoNLL-2017 system on end-to-end universal parsing includes three cascaded modules, *tokenization*, *POS tagging* and dependency parsing. We develop effective neural models for each task, with particular utilization of bidirectional LSTM networks. Furthermore, we use transfer parsing approaches for cross-domain and cross-lingual adaption, that can effectively exploit resources from multiple treebanks. We obtain significant improvements against the UDPipe baseline systems on most of the test sets, and obtain the 4th place in the final evaluation.

ltcode	Supervised		Transfer	
	UAS	LAS	UAS	LAS
uk	78.75	72.47	86.27	80.92
ga	76.66	67.08	80.83	73.44
ug	58.53	38.32	67.19	52.23
Average	*71.31*	*59.29*	***78.10***	***68.86***

Table 6: Effects of cross-lingual transfer parsing on ug uk and ga. 5-fold cross-validation is used for evaluation.

6 Credits

There are a few references we would like to give proper credit, especially to data providers: the core Universal Dependencies paper from LREC 2016 (Nivre et al., 2016), the UD version 2.0 datasets (Nivre et al., 2017b,a), the baseline UDPipe models (Straka et al., 2016b), the baseline SyntaxNet models (Weiss et al., 2015) and the evaluation platform TIRA (Potthast et al., 2014).

Acknowledgments

This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61300113 and 61632011.

References

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proc. of ACL*. pages 1052–1062.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proc. of ACL*. pages 32–37.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL and IJCNLP*. pages 334–343.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *NAACL*. Association for Computational Linguistics, Atlanta, Georgia, pages 644–648.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pages 1234–1244.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *AAAI*. pages 2734–2740.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL* 4:313–327.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Master thesis, Massachusetts Institute of Technology.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR) Workshop* .

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proc. of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. pages 50–57.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Silvie Cinková, Çar Çöltekin, Miriam Connor, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà M, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Phng Lê Hng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins,

ltcode	UDPipe 1.1		Ours		ltcode	UDPipe 1.1		Ours	
	UAS	LAS	UAS	LAS		UAS	LAS	UAS	LAS
ar	71.19	65.30	74.13	69.12	hsb	61.70	53.83	66.64	59.27
ar_pud	53.55	43.14	57.18	48.01	hu	71.46	64.30	74.68	66.29
bg	87.79	83.64	90.30	86.73	id	80.91	74.61	83.06	76.66
bxr‡	46.97	31.50	46.04	27.66	it	88.03	85.28	90.05	87.77
ca	88.62	85.39	90.79	88.27	it_pud	87.04	83.70	88.59	85.51
cs	86.46	82.87	89.57	86.52	ja	73.52	72.21	81.94	80.85
cs_cac	86.49	82.46	87.66	83.87	ja_pud	77.13	76.28	84.38	83.75
cs_cltt	76.26	71.64	84.96	81.89	kk	41.92	24.51	42.11	24.76
cs_pud	84.42	79.80	85.79	80.75	kmr	46.20	32.35	52.55	44.70
cu	69.68	62.76	72.19	65.80	ko	66.40	59.09	76.95	71.82
da	76.94	73.38	81.14	78.03	la	54.35	43.77	59.15	48.75
de	74.27	69.11	79.03	74.79	la_ittb	80.78	76.98	84.07	81.03
de_pud	73.64	66.53	77.90	71.11	la_proiel	63.50	57.54	68.94	63.48
el	83.00	79.26	85.72	82.82	lv	67.14	59.95	71.91	64.97
en	78.87	75.84	82.88	79.94	nl	74.94	68.90	78.90	73.43
en_lines	77.39	72.94	82.70	78.73	nl_lassysmall	81.37	78.15	89.06	86.85
en_partut	77.83	73.64	85.57	81.98	no_bokmaal	86.14	83.27	89.09	86.90
en_pud	82.74	78.95	84.97	81.86	no_nynorsk	84.88	81.56	87.95	85.43
es	84.84	81.47	87.20	84.22	pl	85.08	78.78	88.18	83.75
es_ancora	86.97	83.78	89.94	87.39	pt	85.77	82.11	87.75	84.90
es_pud‡	84.71	77.65	82.34	72.67	pt_br	87.75	85.36	90.51	88.71
et	67.71	58.79	73.09	65.10	pt_pud	80.10	73.96	81.18	72.33
eu	74.39	69.15	79.29	73.85	ro	85.50	79.88	87.30	82.21
fa	83.36	79.24	86.24	82.08	ru	79.28	74.03	84.32	80.58
fi	77.90	73.75	81.98	77.73	ru_pud‡	75.67	68.31	72.33	61.60
fi_ftb	78.77	74.03	82.79	78.08	ru_syntagrus	89.30	86.76	91.71	89.77
fi_pud	82.24	78.65	82.76	78.99	sk	78.14	72.75	84.38	79.82
fr	84.13	80.75	86.07	82.67	sl	84.68	81.15	89.54	87.08
fr_partut	81.69	77.38	88.39	84.86	sl_sst	53.79	46.45	60.36	54.06
fr_pud	78.62	73.63	82.55	77.51	sme	46.06	30.6	52.51	38.91
fr_sequoia	82.62	79.98	87.11	85.09	sv	80.78	76.73	83.93	80.58
ga	72.08	61.52	73.48	61.62	sv_lines	79.18	74.29	81.77	77.30
gl	80.66	77.31	83.31	80.23	sv_pud	75.09	70.62	75.15	70.70
gl_treegal	71.17	65.82	72.65	66.51	tr	60.48	53.19	64.14	56.43
got	67.13	59.81	67.61	60.52	tr_pud‡	55.01	34.53	54.17	34.15
grc	62.74	56.04	66.86	59.84	ug‡	53.58	34.18	51.57	34.52
grc_proiel	70.42	65.22	74.19	69.39	uk	69.78	60.76	71.22	63.08
he	61.54	57.23	64.30	60.07	ur	83.67	76.69	86.41	79.72
hi	90.97	86.77	93.31	89.48	vi	42.12	37.47	47.53	42.52
hi_pud	63.43	50.85	67.24	54.14	zh	61.5	57.40	68.95	65.10
hr	83.20	77.18	86.58	81.30	***Average***	**74.41**	**68.35**	**77.81**	**72.11**

Table 7: End-to-end parsing results on all test treebanks. ‡ indicates the test sets on which UDPipe performs better. Among the 5 sets, es_pud, ru_pud and tr_pud are parallel test sets on which we simply use the model trained from the source treebank. We suggest better strategies should be explored.

ltcode	Ours (b/r)		Ours (a/r)		Rank-1	
	WSeg	LAS	WSeg	LAS	WSeg	LAS
ja	92.95	80.85	94.70	84.37	**98.59**	**91.13**
ja_pud	94.02	83.75	**95.54**	**85.33**	94.93	83.75
vi	84.70	42.52	**91.40**	**48.98**	87.30	47.51
zh	91.19	65.10	**95.21**	**70.49**	94.57	68.56

Table 8: Post-evaluation results on zh, vi, ja and ja_pud. b/r: before revision. a/r: after revision.

Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Rudolf Rosa, Davide Rovati, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Jana Str-

nadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zhuoran Yu, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 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, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà M, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phng Lê Hng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 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.

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.

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 Hajic, and Jana Straková. 2016a. Ud-pipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Milan Straka, Jan Hajič, and Jana Straková. 2016b. 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.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *NAACL*. Association for Computational Linguistics, Montréal, Canada, pages 477–487.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL and IJCNLP*. pages 1556–1566.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2016. Learning distributed word representations for bidirectional lstm recurrent neural network. In *Proc. of HLT-NAACL*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. *CoRR* abs/1506.06158. http://arxiv.org/abs/1506.06158.

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.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proc. of ICML*. pages 1604–1612.