Combining Global Models for Parsing Universal Dependencies

Tianze Shi **Felix G. Wu** **Xilun Chen** **Yao Cheng**
Cornell University
{tianze,felixgwu,xlchen,yc2258}@cs.cornell.edu

Abstract

We describe our entry, C2L2, to the CoNLL 2017 shared task on parsing Universal Dependencies from raw text. Our system features an ensemble of three global parsing paradigms, one graph-based and two transition-based. Each model leverages character-level bi-directional LSTMs as lexical feature extractors to encode morphological information. Though relying on baseline tokenizers and focusing only on parsing, our system ranked second in the official end-to-end evaluation with a macro-average of 75.00 LAS F1 score over 81 test treebanks. In addition, we had the top average performance on the four surprise languages and on the small treebank subset.

1 Introduction

General Parsing Approach Our submitted system to the CoNLL 2017 shared task (Zeman et al., 2017) focuses only on the task of dependency parsing, assuming that tokenization, sentence boundary detection, part-of-speech (POS) tagging and morphological features are already handled by a baseline model. In this paper, we highlight our neural-network-based feature extractors and ensemble of global parsing models, including two novel global transition-based models.

Bi-directional long-short term memory networks (Graves and Schmidhuber, 2005, bi-LSTMs) have recently achieved state-of-the-art performance on syntactic parsing (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016; Dozat and Manning, 2017). Our system leverages the representational power of bi-LSTMs to generate compact features for both graph-based and transition-based parsing frameworks. The latter further enables the application of dynamic programming techniques (Huang and Sagae, 2010; Kuhlmann et al., 2011) for global training and exact decoding. With just two bi-LSTM vectors as features, all three global parsing paradigms in our system have efficient $O(n^3)$ implementations. The full system consists of 3-5 each of these unlabeled parsing models (9-15 in total, depending on the treebank), and another ensemble of arc labelers.

Adaptation of General Approach to the Shared Task The CoNLL 2017 shared task presents two unique challenges: 1. A large fraction of the datasets are morphologically-rich languages. Some languages have an exceedingly-high out-of-vocabulary ratio of over 30%. 2. For many languages, very little training data is provided. Furthermore, there are four surprise language, for which we only have tens of sample sentences.

We address the first challenge with character-level bi-LSTMs, which have previously been shown to be effective in multi-lingual POS tagging (Plank et al., 2016) and dependency parsing (Ballesteros et al., 2015; Alberti et al., 2017). Character-level representation gives better coverage, and it directly learns sub-word information through end-to-end training.

The second challenge is approached by transferring delexicalized information. For each of those languages with little training data, we select the most similar language according to linguistic typology. We then train delexicalized models taking only part-of-speech and morphology tags as input features, which are made available through baseline prediction during test time.

Our full system scored a macro-average LAS F1 score of 75.00, which ranked second among all participating systems. Additionally, in the categories of small treebanks and surprise languages, we obtained the best average performance.

Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 31–39,
Vancouver, Canada, August 3-4, 2017.

2 System Overview

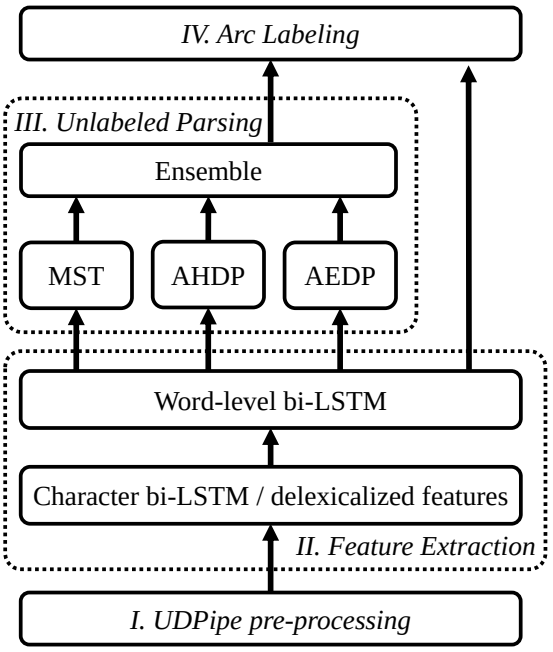

Figure 1: Overview of our system.

Figure 1 illustrates our pipelined system. It processes raw texts in four stages starting from baseline UDPipe (Straka et al., 2016) tokenization and sentence delimitation. For this stage we use predictions provided by the organizers instead of training our own UDPipe models.

For each sentence, Stage II (§3) extracts a dense feature vector for each word in the sentence. For most languages, we employ character-level bi-LSTMs to capture morphological information. On top of the character-level representations, there is another layer of bi-LSTMs processing at the word level, the output of which gives context-sensitive features associated with every word in the sentence. For the four surprise languages and a selected set of languages with small training treebanks, we substitute the character-level encodings of each word in Stage II with concatenation of part-of-speech (POS) tag embeddings and morphological feature embeddings, but keep the word-level bi-LSTMs. We call these *delexicalized* features as opposed to the *lexicalized* features in the general case. All later stages are kept the same. The POS tags and morphological features are provided by baseline UDPipe predictions.

Stage III (§4.1) focuses on unlabeled parsing with an ensemble of three global models, one first-order graph-based maximal spanning tree algorithm (MST), and two transition-based, namely arc-hybrid and arc-eager dynamic programming (AHDP and AEDP). They share the same underlying feature extractors. We combine outputs from the unlabeled parsing models with a uniform weight reparsing model (Sagae and Lavie, 2006).

The final stage (§4.2) of our system is arc labeling. Based on the extracted LSTM features and predicted unlabeled parse trees, this stage assigns the highest scoring label to each arc. Similar to Stage III, we train multiple models with different random initializations, and the ensemble prediction is obtained via majority vote.

Our system was implemented with DyNet (Neubig et al., 2017). Each single model is of small size and runs efficiently. The submitted full system completed the test phase in 4.64 hours with 2 threads. We provide implementation details for all the modules and training process in §6. The code is available at `https://github.com/CoNLL-UD-2017/C2L2`.

3 Feature Extractors

In Stage II of our system, we first extract features for each word in isolation, then consider one sentence at a time for context-sensitive representations. These two feature extractors both leverage the representational power of bi-LSTMs.

3.1 Character LSTMs

Among the most straightforward ways for representing a word are through binary features or word embeddings. Though popular in many existing parsers, they are not ideal for languages with high out-of-vocabulary (OOV) ratios. In Universal Dependencies, the 56 development sets have an average OOV ratio of 14.4%, with four languages (`et`, `hu`, `ko` and `sk`) higher than 30%, posing a severe challenge for lexical representation. On the other hand, the average out-of-charset (OOC) ratio is 0.03%, with the highest (`zh`) not exceeding 0.1%, suggesting the promise of character-level representations in terms of coverage.

Our system adopts character-level bi-LSTMs similar to Plank et al. (2016) and Ballesteros et al. (2015). They show that the obtained sub-word information is especially useful for rare and OOV words in morphologically-rich languages.

Formally, for a word w with its character sequence $[\text{BOW}, c_1, ..., c_m, \text{EOW}]$, with two special begin-of-word (BOW, or c_0) and end-of-word (EOW, or c_{m+1}) symbols, we run a forward and a

backward LSTM at layer l:

$$[\overrightarrow{c_i^l}] = \text{LSTM}_{\text{forward}}([\overrightarrow{c_i^{l-1}}])$$
$$[\overleftarrow{c_i^l}] = \text{LSTM}_{\text{backward}}([\overrightarrow{c_i^{l-1}}])$$
$$\overrightarrow{c_i^l} = \overrightarrow{c_i^l} \circ \overleftarrow{c_i^l}$$

each c_i^l denotes the vector representation at layer l for c_i, $\circ$ denotes concatenation of vectors, and $[\cdot]$ is a shorthand for a list of vectors. The inputs to the first layer c_i^0 are character embeddings that are jointly trained with the model. We take the concatenation of $\overrightarrow{c_{m+1}}$ and $\overleftarrow{c_0}$ at the final layer of the LSTMs as the output vectors. We use two-layer bi-LSTMs in our system.

Efficiency Improvement Considering the Zipfian distribution for word frequencies, most of the time is spent on getting char bi-LSTM representations for frequent words. On the other hand, for those words, it is considerably easier to train decent representations even without char bi-LSTMs. We thus directly learn the dense word vectors for frequent words, as a proxy for character-level bi-LSTMs and they can be considered as fast look-up tables without actually running the LSTMs[1] .

3.2 Delexicalized Features

For languages with small treebanks, the provided data is not adequate to learn character bi-LSTMs. We choose to use the available delexicalized information predicted by UDPipe. Namely, we use information from two fields: universal POS tags (UPOS) and morphological tags.

To get dense vectors for each word w in the same form as the output of char bi-LSTMs, we use the concatenation of UPOS embeddings $\overrightarrow{p_w}$ and the bag-of-morphology (BOM) embeddings $\text{pool}(\{\overrightarrow{m_w}\})$. The BOM embeddings require a pooling function $\text{pool}(\cdot)$ because each word may receive multiple morphological tags. In our system, we use element-wise max operator as the pooling function.

3.3 Word-level LSTMs

The character bi-LSTM vector for each word is computed in isolation from other words in the sentence. In this module, we again leverage bi-LSTMs for integration of contextual information.

[1] In retrospect, we could have used pre-trained word vectors as extra features.

Similar to §3.1, we pad a sentence with two special begin-of-sentence (BOS, or w_0), and end-of-sentence (EOS, or w_{n+1}) symbols into $[\text{BOS}, w_1, ..., w_n, \text{EOS}]$. Inputs to the first layer are character bi-LSTM encodings, or concatenation of POS-tag and BOM embeddings in the case of delexicalized models. We take the bi-directional vectors $\overrightarrow{w_i}$ at the final layer as the context-sensitive representation associated with w_i. All parsing components to be described in the following section will build from these vectors.

4 Parsing Components

Our system parses a sentence in two steps, first predicting the unlabeled parse tree, and next predicting the label for each arc in the unlabeled tree.

4.1 Global Models for Unlabeled Parsing

Our system includes one graph-based and two transition-based, a total of three different global parsing paradigms. All of these models only handle projective cases. For this reason, before training, we projectivize all gold-standard trees in the training sets.

First-order Graph-based Parsing Our graph-based model is based on the popular edge-factored Eisner's algorithm (Eisner, 1996; Eisner and Satta, 1999). Each potential arc (h, m) in the graph ($O(n^2)$ in total with sentence length n) is first scored with a function $\text{score}^{\text{MST}}(h, m)$. Then Eisner's algorithm is used to find the maximum spanning tree among all possible projective trees:

$$\underset{\text{valid parses } y}{\text{argmax}} \sum_{(h,m)\in y} \text{score}^{\text{MST}}(h, m)$$

Following Dozat and Manning (2017), we use a deep bi-affine scoring function:

$$\text{score}^{\text{MST}}(h, m) = v_h^{\intercal} U v_m + b_h \cdot v_h + b_m \cdot v_m + b$$

where

$$v_h = \text{MLP}^{\text{MST-head}}(\overrightarrow{h})$$
$$v_m = \text{MLP}^{\text{MST-mod}}(\overrightarrow{m})$$

are representations transformed by two multi-layer perceptrons (MLPs) from their bi-LSTM vectors. We train separate MLPs for head and modifier transformation. The weight matrix U, bias vectors b_h, b_m and term b are parameters of the function.

Global Transition-based Parsing We include global training and exact decoders for two transition systems, arc-hybrid and arc-eager. They are based on dynamic programming approaches (Huang and Sagae, 2010; Kuhlmann et al., 2011), thus we call the two models AHDP and AEDP.

The dynamic programming shares computation for parser configurations with the same extracted features. In our system, we only use two bi-LSTM vectors, one from the top of the stack ($\overleftarrow{s_0}$), and one from the top of the buffer ($\overleftarrow{b_0}$). This compact set of features enables dynamic programming to compress the exponentially-large search space down to $O(n^3)$ for the two transition systems.

Below we illustrate the AHDP decoder, with AEDP being similar. The bare deduction system, adapted from Kuhlmann et al. (2011) is:

$$\mathsf{sh}\ \frac{[i,j]}{[j,j+1]} \qquad \mathsf{re}_{\curvearrowright}\ \frac{[k,i] \quad [i,j]}{[k,j]} k \curvearrowright i$$

$$\mathsf{re}_{\curvearrowleft}\ \frac{[k,i] \quad [i,j]}{[k,j]} i \curvearrowleft j$$

each deduction item $[i,j]$ corresponds to a push computation detailed in Kuhlmann et al. (2011). For the purpose of our decoder, the deduction item can also be understood as a parser configuration with w_i being s_0 and w_j being b_0. The deduction system has an axiom $[0,1]$ and goal $[0,n+1]$ corresponding to initial and terminal configurations.

Next, we incorporate scoring functions:

$$\frac{[i,j]:v}{[j,j+1]:0}(\mathsf{sh}) \quad \frac{[k,i]:v_1 \quad [i,j]:v_2}{[k,j]:v_1+v_2+\Delta}(\mathsf{re}_{\curvearrowright})$$

where $\Delta = \text{score}_{\mathsf{sh}}(\overleftarrow{w_k}, \overleftarrow{w_i}) + \text{score}_{\mathsf{re}_{\curvearrowright}}(\overleftarrow{w_i}, \overleftarrow{w_j})$. The scoring functions are bi-affine and take the same form as $\text{score}^{\text{MST}}(\cdot)$. The highest-scoring proof for the goal item $[0, n+1]$ constitutes the predicted transition sequence.

Training We employ discriminative training strategies for all three global parsing models. Cost-augmented decoding (Taskar et al., 2005; Smith, 2011) is applied during training. A correct parse tree is instructed to get higher scores than an incorrect parse tree by a margin set to be the number of incorrectly-attached nodes (Hamming distance). This technique has previously been applied in training a neural MST parser (Kiperwasser and Goldberg, 2016).

	UAS F1	LAS F1	Official Ranking
Big Treebanks	85.16	79.85	2
Small Treebanks	70.59	61.49	1
PUD Treebanks	80.17	71.49	2
Surprise Languages	58.40	47.54	1
Overall	80.35	75.00	2

Table 1: Official UAS and LAS scores on the test sets. Rankings are based on the macro-average LAS F1 scores over all treebanks in the set.

Target	Source	UAS F1	LAS F1	Official Ranking
bxr	hi	50.79	31.98	2
hsb	cs	69.45	61.70	1
kmr	fa	54.51	47.53	1
sme	fi	58.85	48.96	1
Average		58.40	47.54	1

Table 2: Evaluation results of our system on the surprise languages. We show the source treebanks from which we trained the delexicalized parsers.

4.2 Arc Labeling

We separate out the stage of arc labeling and adopt a simple labeler proposed by Kiperwasser and Goldberg (2016). For a predicted arc with h as the head and m being the modifier, their associated vectors are concatenated to be the input to a MLP. Each dimension of the output from the MLP corresponds to the score for a potential label, And we select the label with the highest score:

$$\text{label}(h,m) = \underset{l}{\text{argmax}}\, \text{MLP}^{\text{label}}_l(\overleftarrow{h} \circ \overleftarrow{m})$$

The arc-labeling models are trained with gold-standard (h,m) tuples. And we use a discriminative hinge loss, with margin of 1.

5 Results

The main official evaluation results are given in Table 1. Our system achieved second place in overall ranking. When considering average performance on small treebanks (8 treebanks) and surprise languages (4 treebanks, detailed in Table 2), we scored the first among all teams.

We show per-treebank LAS F1 results in Figure 2. Our system lacks customized modules

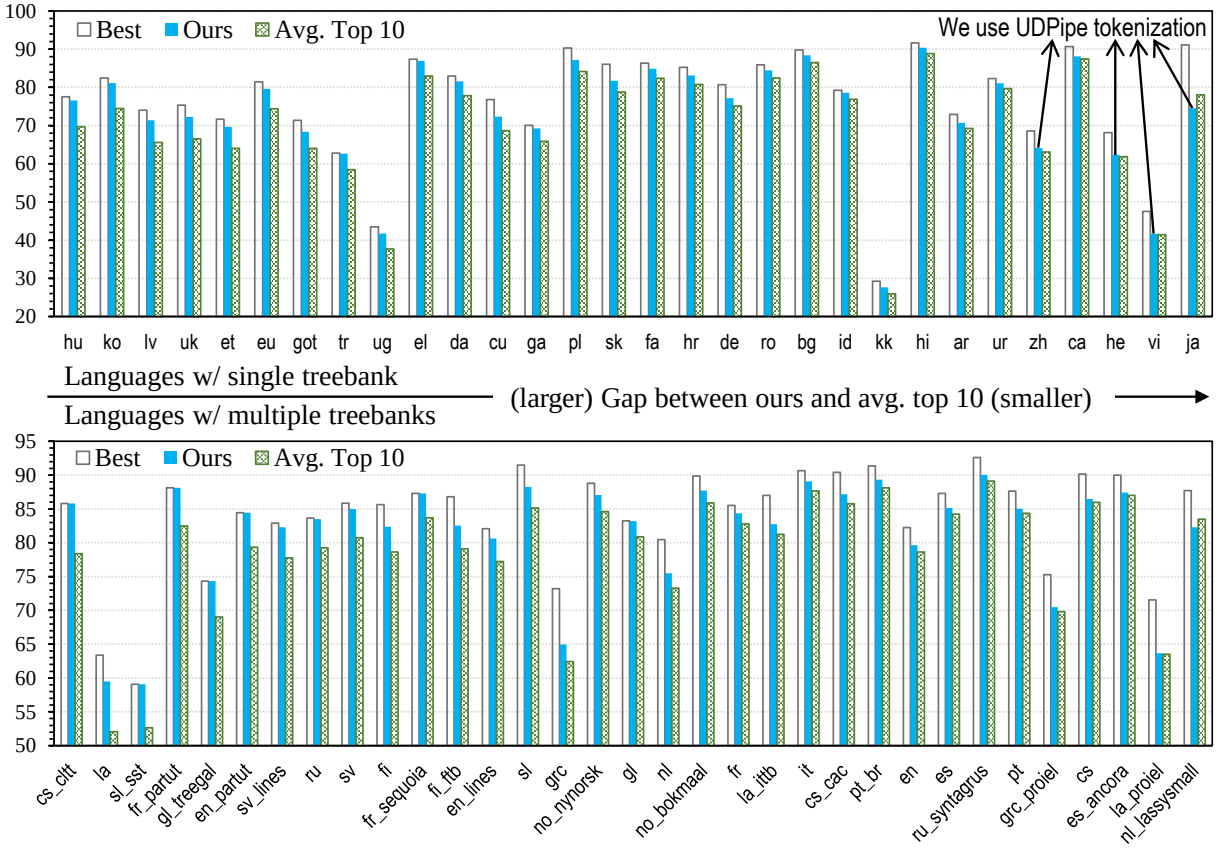

Figure 2: LAS F1 score per treebank. The top/bottom row results are on languages with single/multiple treebank(s). For comparison, we include the best official result and the average of the top ten results on each treebank. Each row is sorted by the gap between our system and the average of the top ten.

for tokenization and sentence boundary detection, which is reflected by the gap between our system and the best-performing systems on `ja`, `vi`, `he` and `zh`. The other large source of gaps comes from languages with large non-projective ratios, such as `grc`, `la` and `nl`. The global transition-based AHDP and AEDP models are not compatible with non-projective parsing, and we did not implement or test with non-projective graph-based parsers due to time and resource constraints.

Our system performs relatively well on languages with high OOV ratios, such as `hu`, `ko`, `lv` and `et`, with the help of character bi-LSTMs. In addition, the strategies of concatenating multiple training treebanks for the same language (see §6) brought success on small treebanks.

Table 3 gives the performance of our system on the 14 additional parallel treebanks. The results are largely consistent with in-domain evaluation results, and we ranked within top third for most treebanks except `ja_pud`, `en_pud` and `ru_pud`. We did not implement our own tokenizer for Japanese, explaining the gap. For the other two languages, our selected models were not domain-robust. We perform a post-evaluation analysis and parse the PUD treebanks (Nivre et al., 2017a) with models trained on the canonical treebanks. The two languages observe an improvement on LAS scores of 7.53 and 14.73 respectively.

Ablation Analysis To examine the effect of individual components in our ensemble system, we evaluate several variations, where we use single or an incomplete set of models for unlabeled parsing and arc-labeling. Results are shown in Table 4. AEDP gives higher unlabeled parsing performance, and an ensemble of three instances of AEDPs achieves comparable performance to our full system. The arc-labeling ensemble gives another gain in LAS result of 0.31.

6 Implementation Details

Our system was trained on the UD 2.0 dataset (Nivre et al., 2016, 2017b), with the provided training and development splits when available. For languages without development sets, we split

Target Treebank	Selected Model	LAS F1	Rank
pt_pud	pt	78.48	1
de_pud	de	73.92	2
sv_pud	sv	77.97	2
fr_pud	fr	78.25	2
es_pud	es	80.50	2
fi_pud	fi	85.42	2
it_pud	it	86.74	2
tr_pud	tr	37.65	3
ar_pud	ar	49.03	3
hi_pud	hi	54.12	3
cs_pud	cs_cac	82.23	3
cs_pud	cs*	83.38*	2*
ja_pud	ja	78.22	6
ru_pud	ru_syntagrus	61.82	22
ru_pud	ru*	76.55*	1*
en_pud	en_lines	76.56	23
en_pud	en*	84.09*	2*
Average		71.49	2

Table 3: Evaluation results of our system on PUD treebanks. We give post-evaluation (non-official) results* where we tested with models trained on treebanks with canonical language codes. The table is sorted by our rankings.

Unlabeled Parser	Arc Labeler	LAS F1
Full	Full	75.00
3×AEDP	Full	74.79
Full	Single	74.69
1×AEDP	Full	74.32
1×AHDP	Full	74.00
1×MST	Full	73.75

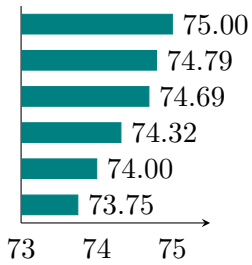

Table 4: Ablation of our ensemble system.

the training sets into train/dev sets with ratio 0.9/0.1. We did not use any additional data. All neural network computation was implemented with DyNet (Neubig et al., 2017).

Stage I of our system is the baseline system UDPipe 1.1, and we directly used the outputs provided by the organizers. We implemented modules for all later stages. They were trained with gold-standard features and tokenizations. For all languages and all treebanks, we trained models with 2-layer-deep and 192-unit-wide (96 units for each direction) word-level bi-LSTMs as feature extractors. Lexicalized character bi-LSTMs are 2 layers deep and 128 units wide, with 64-dimensional input character embeddings. For languages without lexicalized feature extractors, we used concatenation of 64-dimensional UPOS embeddings, and max pooling of 64-dimensional morphological embeddings as input to word-level bi-LSTMs.

The word-level bi-LSTM feature vectors were passed through MLPs with 1 hidden layer and 192 hidden units, before the bi-affine scoring functions for MST, AHDP and AEDP unlabeled parsing. In arc-labelers, we concatenated the word-level feature vectors and passed it through a 1-layer MLP with 192 hidden units to get scores for the arc labels. Output layer size depends on the number of labels appearing in the training set for the concerned treebank. We projected language-specific arc tags into universal ones before training.

All the aforementioned hidden layers used tanh as activation functions. And the parameters were uniformly initialized (Glorot and Bengio, 2010), except for the weight matrices in the bi-affine scoring functions, which were initialized to be orthogonal (Saxe et al., 2013). We did not use any pre-trained word embeddings.

We applied dropout at every stage. MLPs had dropout rates of 0.3 (Srivastava et al., 2014). Bi-LSTMs, both character-level and word-level, also had dropout rates of 0.3 for input and recurrent connections (Gal and Ghahramani, 2016). Further, we zeroed out input vectors to word-level LSTMs for 15% of the time, to encourage the models gain more information from context.

When we trained each model, we randomly shuffled the training set before starting each epoch, and grouped sentences into mini-batches of approximately 100 words. The discriminative loss functions were optimized via Adam optimizer (Kingma and Ba, 2015), with default hyperparameters except initial learning rate set to be 0.002. We evaluated the models with development data after every 500 mini-batches. We halved the learning rate if the performance plateaued in 5 consecutive evaluations, The process was repeated 3 times before we terminated the training process.

We employed the technique of stack-propagation (Zhang and Weiss, 2016), where the auxiliary task of UPOS prediction was used as a regularizer. It received 0.1 the weight of other

components in computing the loss.

For the languages with multiple treebanks, we first concatenated the training treebanks and trained a general model. We then fine-tuned the models on the respective individual treebanks.

To speed up training,we simultaneously trained MST, AHDP, AEDP and arc labeling models with shared LSTM feature extractors. Their losses were linearly combined with weights 0.6, 0.3, 0.3, 1.5 respectively. After a joint model had been trained, we fine-tuned each of the four tasks separately.

Our final system included ensembles both for unlabeled parsing and arc labeling. They were obtained with different random initializations of the neural network, but trained in the same fashion. For languages with multiple treebanks, we trained 3 sets of models (3 for each parsing paradigm, 9 unlabeled parsing models in total, plus 3 for arc labeling). For languages with single treebanks, we trained 5 sets of models.

For surprise languages, we first trained delexicalized models using the training data in a most similar language according to the WALS features (Dryer and Haspelmath, 2013). We selected `fi`, `fa`, `hi`, `cs` for `sme`, `kmr`, `bxr`, `hsb` respectively. We then fine-tuned the models on the sample data for these languages. We treated `kk` and `ug` similarly as they have quite small training sets. Both of them used `tr` as the source language.

The entire training process of all models in the ensemble for all treebanks was done using 8 CPU cores (2 $\times$ Intel i7-4790 @ 3.60GHz) in approximately one week. Each model required at most 2GB RAM plus the amount needed for holding the training sets. On the online evaluation platform TIRA (Potthast et al., 2014), the test phase for our full model finished in 4.64 hours with 2 threads. Each model required at most 500MB RAM plus the amount needed for holding the test sets.

Acknowledgments

The first author was supported by a Google focused research grant. The second author was supported by Kilian Q. Weinberger with IIS-1550179, IIS-1525919, IIS-1618134 grants from National Science Foundation. The fourth author was supported by DARPA DEFT Grant FA8750-13-2-0015. We thank Lillian Lee for her helpful input and support throughout the shared task. And we thank the two anonymous reviewers for their valuable comments.

References

Chris Alberti, Daniel Andor, Ivan Bogatyy, Michael Collins, Dan Gillick, Lingpeng Kong, Terry Koo, Ji Ma, Mark Omernick, Slav Petrov, Chayut Thanapirom, Zora Tung, and David Weiss. 2017. SyntaxNet models for the CoNLL 2017 shared task. *arXiv:1703.04929*. http://arxiv.org/abs/1703.04929.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 349–359. https://doi.org/10.18653/v1/D15-1041.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 32–37. https://doi.org/10.18653/v1/P16-2006.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*. http://aclweb.org/anthology/C96-1058.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. pages 457–464. http://aclweb.org/anthology/P99-1059.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1019–1027.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. volume 9, pages 249–256.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(56):602–610. https://doi.org/10.1016/j.neunet.2005.06.042.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, pages 1077–1086. http://aclweb.org/anthology/P10-1110.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 4th International Conference on Learning Representations*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. http://aclweb.org/anthology/Q16-1023.

Marco Kuhlmann, Carlos Gómez-Rodrìguez, 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*. Association for Computational Linguistics, pages 673–682. http://aclweb.org/anthology/P11-1068.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv:1701.03980*. http://arxiv.org/abs/1701.03980.

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*. European Language Resources Association, Portoro, Slovenia, pages 1659–1666.

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.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 412–418. https://doi.org/10.18653/v1/P16-2067.

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*. Springer, Berlin Heidelberg New York, pages 268–299. https://doi.org/10.1007/978-3-319-11382-1_22.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, Companion Volume: Short Papers*. pages 129–132. http://aclweb.org/anthology/N06-2033.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Representations*.

Noah A. Smith. 2011. *Linguistic Structure Prediction*, volume 4 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

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*. European Language Resources Association, Portoro, Slovenia.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*. pages 896–903.

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.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1557–1566. https://doi.org/10.18653/v1/P16-1147.