From Raw Text to Universal Dependencies – Look, No Tags!

Miryam de Lhoneux* **Yan Shao*** **Ali Basirat*** **Eliyahu Kiperwasser†**
Sara Stymne* **Yoav Goldberg†** **Joakim Nivre***

*Department of Linguistics and Philology
Uppsala University
Uppsala, Sweden

†Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel

Abstract

We present the Uppsala submission to the CoNLL 2017 shared task on parsing from raw text to universal dependencies. Our system is a simple pipeline consisting of two components. The first performs joint word and sentence segmentation on raw text; the second predicts dependency trees from raw words. The parser bypasses the need for part-of-speech tagging, but uses word embeddings based on universal tag distributions. We achieved a macro-averaged LAS F1 of 65.11 in the official test run, which improved to 70.49 after bug fixes. We obtained the 2nd best result for sentence segmentation with a score of 89.03.

1 Introduction

The CoNLL 2017 shared task differs from most previous multilingual dependency parsing tasks not only by using cross-linguistically consistent syntactic representations from the UD project (Nivre et al., 2016), but also by requiring systems to start from raw text, as opposed to pre-segmented and (often) pre-annotated words and sentences. Since systems are only evaluated on their output dependency trees (and indirectly on the word and sentence segmentation implicit in these trees), developers are free to choose what additional linguistic features (if any) to predict as part of the parsing process.

The Uppsala team has adopted a minimalistic stance in this respect and developed a system that does not predict any linguistic structure over and above a segmentation into sentences and words and a dependency structure over the words of each sentence. In particular, the system makes no use of part-of-speech tags, morphological features, or lemmas, despite the fact that these annotations are available in the training and development data.

In this way, we go against a strong tradition in dependency parsing, which has generally favored pipeline systems with part-of-speech tagging as a crucial component, a tendency that has probably been reinforced by the widespread use of data sets with gold tags from the early CoNLL tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). Even models that perform joint inference, like those of Hatori et al. (2012) and Bohnet et al. (2013), depend heavily on part-of-speech tags, so we were unlikely to reach top scores in the shared task without them. However, from a scientific perspective, we thought it would be interesting to explore how far we can get with a bare-bones system that does not predict redundant linguistic categories. In addition, we take inspiration from recent work showing that character-based representations can at least partly obviate the need for part-of-speech tags (Ballesteros et al., 2015).

The Uppsala system is a very simple pipeline consisting of two main components. The first is a model for joint sentence and word segmentation, which uses the BiRNN-CRF framework of Shao et al. (2017) to predict sentence and word boundaries in the raw input and simultaneously marks multiword tokens that need non-segmental analysis. The latter are handled by a simple dictionary lookup or by an encoder-decoder network. We use a single universal model regardless of writing system, but train separate models for each language. The segmentation component is described in more detail in Section 2.

The second main component of our system is a greedy transition-based parser that predicts the dependency tree given the raw words of a sentence. The starting point for this model is the transition-based parser described in Kiperwasser and Goldberg (2016b), which relies on a BiLSTM to learn

informative features of words in context and a feed-forward network for predicting the next parsing transition. The parser uses the arc-hybrid transition system (Kuhlmann et al., 2011) with greedy inference and a dynamic oracle for exploration during training (Goldberg and Nivre, 2013). For the shared task, the parser has been modified to use character-based representations instead of part-of-speech tags and to use pseudo-projective parsing to capture non-projective dependencies (Nivre and Nilsson, 2005). The parsing component is further described in Section 3.

Our original plans included training a single universal model on data from all languages, with cross-lingual word embeddings, but in the limited time available we could only start exploring two simple enhancements. First, we constructed word embeddings based on the RSV model (Basirat and Nivre, 2017), using universal part-of-speech tags as contexts (Section 4). Secondly, we used multilingual training data for languages with little or no training data (Section 5).

Our system was trained only on the training sets provided by the organizers (Nivre et al., 2017a). We did not make any use of large unlabeled data sets, parallel data sets, or word embeddings derived from such data. After evaluation on the official test sets (Nivre et al., 2017b), run on the TIRA server (Potthast et al., 2014), the Uppsala system ranked 23 of 33 systems with respect to the main evaluation metric, with a macro-average LAS F1 of 65.11. We obtained the 2nd highest score for sentence segmentation overall (89.03), and top scores for word segmentation on several languages (but with relatively high variance).

However, after the test phase was concluded, we discovered two bugs that had affected the results negatively. For comparison, we therefore also include post-evaluation results obtained after eliminating the bugs but without changing anything else. This gives us a macro-average LAS F1 of 70.49 and a top ten position in the post-evaluation ranking. We discuss our results in Section 6 and refer to the shared task overview paper (Zeman et al., 2017) for a thorough description of the task and an overview of the results.

2 Sentence and Word Segmentation

We model joint sentence and word segmentation as a character-level sequence labeling problem in a Bi-RNN-CRF model (Huang et al., 2015; Ma and Hovy, 2016). We simultaneously predict sentence boundaries and word boundaries and identify multi-word tokens that require further transduction.

In the BiRNN-CRF architecture, characters – regardless of writing system – are represented as dense vectors and fed into the bidirectional recurrent layers. We employ the gated recurrent unit (GRU) (Cho et al., 2014) as the basic recurrent cell. Dropout (Srivastava et al., 2014) is applied to the output of the recurrent layers, which are concatenated and passed further to the first order chain CRF layer. The CRF layer models conditional scores over all possible boundary tags given the features extracted by the BiRNN from the vector representations of the input characters. Incorporating the transition scores between the successive labels, the optimal sequence of labels that indicate different types of boundaries can be obtained efficiently via the Viterbi algorithm.

As illustrated in Figure 1, following Shao et al. (2017), we employ the boundary tags B, I, E, and S to indicate a character positioned at the beginning (B), inside (I), or at the end (E) of a word, or occurring as a single-character word (S). To this standard tag set, we add four corresponding tags (K, Z, J, D) to label corresponding positions in multi-word tokens, and a special tag X to mark characters, mostly spaces, that do not belong to words/tokens. Finally, we mark a character that occurs at the end of a sentence. T is employed if the character is a single-character word and U is used otherwise.

Multi-word tokens are transcribed without considering contextual information. For most languages, the number of unique multi-word tokens is rather limited and can be covered by dictionaries built from the training data. However, if there are more than 200 unique multi-word tokens contained in the training data, we employ an attention-based encoder-decoder (Bahdanau et al., 2014) equipped with shared long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) as the basic recurrent cell. At test time, multi-word tokens are first queried in the dictionary. If not found, the segmented words are generated via the encoder-decoder as a sequence-to-sequence transduction.

Table 1 shows the hyper-parameters adopted for the main network as well as the encoder-decoder, which is trained separately from the main net-

Characters:	... La sede del condado es Ottawa. En ...
Tags:	... BEXBIIEXKZJXBIIIIIEXBEXBIIIIETXBE ...

Figure 1: Tags employed for sentence and word segmentation. Note that the token `del` is a multiword token that should be transcribed to `de` and `el` and should therefore be tagged `KZJ` instead of `BIE`.

Character embedding size	50
GRU/LSTM state size	200
Optimizer	Adagrad
Initial learning rate (main)	0.1
Decay rate	0.05
Gradient Clipping	5.0
Initial learning rate (encoder-decoder)	0.3
Dropout rate	0.5
Batch size	10

Table 1: Hyper-parameters for segmentation.

work. We use one set of parameters for all treebanks. The weights of the neural networks, including the embeddings, are initialized using the scheme introduced in Glorot and Bengio (2010). The network is trained using back-propagation, and all embeddings are fine-tuned during training by back-propagating gradients. Adagrad (Duchi et al., 2011) with mini-batches is employed for optimization. The initial learning rate η_0 is updated with a decay rate ρ as $\eta_t = \frac{\eta_0}{\rho(t-1)+1}$ when training the main network, where t is the index of the current epoch. We use early stopping (Yao et al., 2007) with respect to the performance of the model on the validation sets. For the encoder-decoder, 5% of the training data is randomly subtracted for validation. The score is calculated via how many outputs exactly match the references. For the main network, the F1-score is employed to measure the performance of the model after each epoch during training on the development set.

The general segmentation model is applied to all languages with small variations for Chinese and Vietnamese. For Chinese, the concatenated trigram model introduced in Shao et al. (2017) is employed. For Vietnamese, we first separate punctuation marks and then use space-delimited units as the basic elements for boundary prediction.

Bug in test results: After the official evaluation, we discovered a bug in the segmenter, which affected words and punctuation marks immediately before sentence boundaries. After fixing the bugs, both word segmentation and sentence segmentation results improved, as seen from our post-evaluation results included in Section 6.

3 Dependency Parsing

The transition-based parser from Kiperwasser and Goldberg (2016b) uses a configuration containing a buffer B, a stack Σ, and a set of arcs A. In the initial configuration, all words from the sentence plus a root node are in the buffer and the arc set is empty. A terminal configuration has a buffer with just the root and an empty stack, and the arc set then forms a tree spanning the input sentence. Parsing consists in performing a sequence of transitions from the initial configuration to the terminal one, using the arc-hybrid transition system, which allows three types of transitions, SHIFT, LEFT-ARC$_d$ and RIGHT-ARC$_d$, defined as in Figure 2.

The LEFT-ARC$_d$ transition removes the first item on top of the stack (i) and attaches it as a modifier to the first item of the buffer j with label d, adding the arc (j, d, i). The RIGHT-ARC$_d$ transition removes the first item on top of the stack (j) and attaches it as a modifier to the next item on the stack (i), adding the arc (i, d, j). The SHIFT transition moves the first item of the buffer i to the stack. To conform to the constraints of UD representations, we have added a new precondition to the LEFT-ARC$_d$ transition to ensure that the special root node has exactly one dependent. Thus, if the potential head i is the root node, LEFT-ARC$_d$ is only permissible if the stack contains exactly one element (in which case the transition will lead to a terminal configuration). This precondition is applied only at parsing time and not during training.

A configuration c is represented by a feature function $\phi(\cdot)$ over a subset of its elements and for each configuration, transitions are scored by a classifier. In this case, the classifier is a multi-layer perceptron (MLP) and $\phi(\cdot)$ is a concatenation of BiLSTM vectors on top of the stack and the beginning of the buffer. The MLP scores transitions together with the arc labels for transitions that involve adding an arc (LEFT-ARC$_d$ and RIGHT-ARC$_d$). For more details, see Kiperwasser and

Initialization:	$c_0(x = (w_1, \ldots, w_n)) = ([\,], [1, \ldots, n, 0], \emptyset)$	
Termination:	$C_t = \{c \in C \mid c = ([\,], [0], A)\}$	
Transition		**Condition**
LEFT-ARC$_d$	$(\sigma \vert i, j \vert \beta, A) \Rightarrow (\sigma, j \vert \beta, A \cup \{(j, d, i)\})$	$j \neq 0 \vee \sigma = [\,]$
RIGHT-ARC$_d$	$(\sigma \vert i \vert j, \beta, A) \Rightarrow (\sigma \vert i, \beta, A \cup \{(i, d, j)\})$	
SHIFT	$(\sigma, i \vert \beta, A) \Rightarrow (\sigma \vert i, \beta, A)$	$i \neq 0$

Figure 2: Transitions for the arc-hybrid transition system with an artificial root node (0) at the end of the sentence. The stack Σ is represented as a list with its head to the right (and tail σ) and the buffer B as a list with its head to the left (and tail β).

Goldberg (2016b).

The main modification of the parser for the shared task concerns the construction of the BiLSTM vectors, where we remove the reliance on part-of-speech tags and instead add character-based representations. For an input sentence of length n with words $w_1, \ldots, w_n$, we create a sequence of vectors $x_{1:n}$, where the vector x_i representing w_i is the concatenation of a word embedding, a pretrained embedding, and a character vector. We construct a character vector $ch_e(w_i)$ for each w_i by running a BiLSTM over the characters ch_j ($1 \leq j \leq m$) of w_i:

$$ch_e(w_i) = \text{BILSTM}(ch_{1:m})$$

As in the original parser, we also concatenate these vectors with pretrained word embeddings $pe(w_i)$. The input vectors x_i are therefore:

$$x_i = e(w_i) \circ pe(w_i) \circ ch_e(w_i)$$

Our pretrained word embeddings are further described in Section 4. A variant of word dropout is applied to the word embeddings, as described in Kiperwasser and Goldberg (2016a), and we apply dropout also to the character vectors.

Finally, each input element is represented by a BiLSTM vector, v_i:

$$v_i = \text{BILSTM}(x_{1:n}, i)$$

For each configuration c, the feature extractor concatenates the BiLSTM representations of core elements from the stack and buffer. Both the embeddings and the BiLSTMs are trained together with the model. The model is represented in Figure 3.

With the aim of training a multilingual parser, we additionally created a variant of the parser which adds a language embedding to input vectors in a spirit similar to what is done in Ammar et al. (2016). In this setting, the vector for each word x_i is the concatenation of a word embedding, a pretrained word embedding, a character vector, and a language embedding l_i with the language corresponding to the word. As was mentioned in the introduction, our experiments with this model was limited to the languages with little data. Those experiments are described in Section 5.

$$x_i = e(w_i) \circ pe(w_i) \circ ch_e(w_i) \circ e(l_i)$$

The final change we made to the parser was to use pseudo-projective parsing to deal with non-projective dependencies. Pseudo-projective parsing, as described in Nivre and Nilsson (2005), consists in a pre-processing and a post-processing step. The pre-processing step consists in projectivising the training data by reattaching some of the dependents and the post-processing step attempts to deprojectivise trees in output parsed data. In order for information about non-projectivity to be recoverable after parsing, when projectivising, arcs are renamed to encode information about the original parent of dependents which get re-attached. We used MaltParser (Nivre

Internal word embedding dimension	100
Pre-trained word embedding dimension	50
Character embedding dimension	12
Character BI-LSTM Dimensions	100
Hidden units in MLP	100
BI-LSTM Layers	2
BI-LSTM Dimensions (hidden/output)	200 / 200
α (for word dropout)	0.25
Character dropout	0.33
p_{agg} (for exploration training)	0.1

Table 2: Hyper-parameter values for parsing.

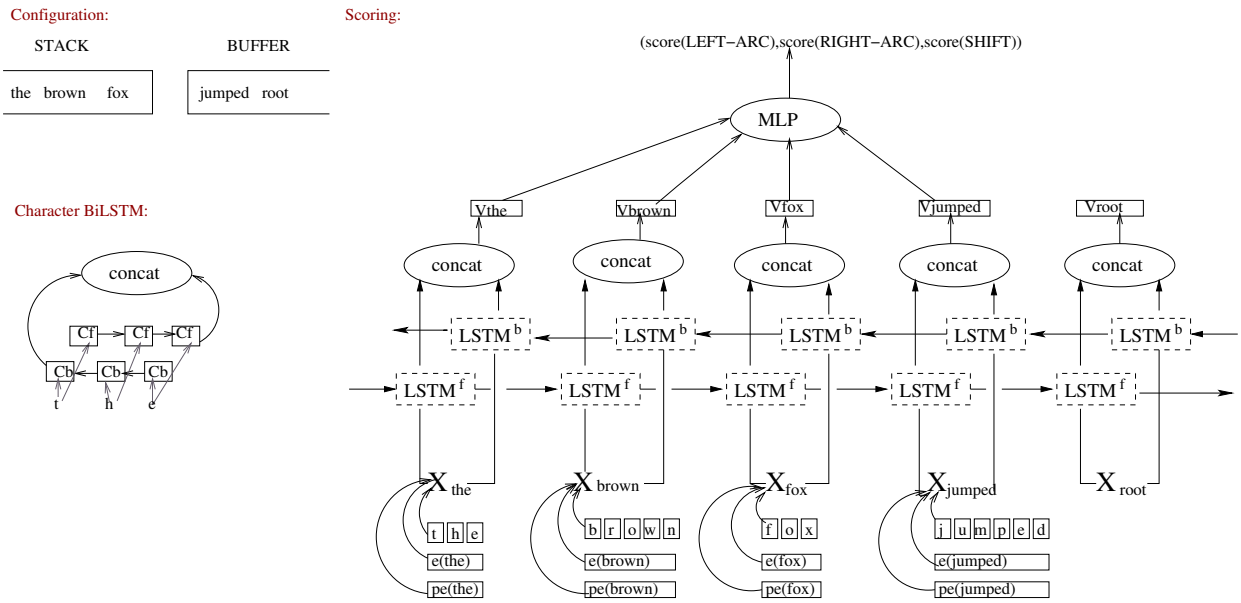

Figure 3: Illustration of the neural model scheme of the transition-based parser when calculating the scores of the possible transitions in a given configuration. The configuration (stack and buffer) is depicted in the top left corner. Each transition is scored using an MLP that is fed the vectors of the first word in the buffer and the three words at the top of the stack, and a transition is picked greedily. Each vector is a BiLSTM encoding of the word. Each x_i is a concatenation of a word vector, a character vector, and an additional external embedding vector for the word. Character vectors are obtained using a BiLSTM over the characters of the word. An example is given at the bottom left of the figure. The figure depicts a single-layer BiLSTM, while in practice we use two layers. When parsing a sentence, we iteratively compute scores for all possible transitions and apply the best scoring action until the final configuration is reached.

et al., 2006) for this. More specifically, we used the *head* schema, as described in Nivre and Nilsson (2005). This method increases the size of the dependency label set. In order to keep training efficient, we cap the number of dependency relations to the 200 most frequently occurring ones in the training set.

We did no hyper-parameter tuning for the parser component but instead mostly used the values that had been found to work well in Kiperwasser and Goldberg (2016b), except for the BiLSTM hidden layer which we increased as we had increased the dimensions of the output layer by using pseudo-projective parsing. The hyper-parameter values we used are in Table 2. We used the dynamic oracle as well as the extended feature set (the top 3 items on the stack together with their rightmost and leftmost modifiers as well as the first item on the buffer and its leftmost modifier). We trained the parsers for 30 epochs and picked the model that gave the best LAS score on the development sets for languages for which we had a development set, the last epoch otherwise.

The code is available at `https://github.com/UppsalaNLP/uuparser`.

Bug in test results: Our official test run suffered from a bug in the way serialization is handled in dynet. As reported in https://github.com/clab/dynet/issues/84, results may differ if the machine on which a model is used does not have the exact same version of boost as the machine on which the model was trained. Our post-evaluations results were obtained by using exactly the same models but parsing the test data on the machine on which they were trained.

4 Pre-Trained Word Embeddings

Our word embedding method is based on the RSV method introduced by Basirat and Nivre (2017). RSV extracts a set of word vectors in three main steps. First it builds a co-occurrence matrix for words that appear in certain contexts. Then, it normalizes the data distribution in the co-occurrence matrix by a power transformation. Finally, it builds a set of word vectors from the singular vectors of the transformed co-occurrence matrix.

We propose to restrict the contexts used in RSV to a set of universal features provided by the UD corpora. The universal features can be any combination of universal POS tags, dependency relations, and other universal tags associated with words. Given the set of universal features, each

word is associated with a high-dimensional vector whose dimensions correspond to the universal features. The space formed by these vectors can be seen as a multi-lingual syntactic space which captures the universal syntactic properties provided by the UD corpora.

We define the set of universal features as $\{t_w, t_h, (t_w, d, t_h)\}$, where t_w and t_h are the universal POS tags of the word of interest and its parent in a dependency tree, and d is the dependency relation between them. It results in a set of *universal word vectors* with fairly large dimensions, 13 794. The values of the vector elements are set with the probability of seeing each universal feature given the word. These vectors are then centered around their mean and the final word vectors are built from the top k right singular vectors of the matrix formed by the high-dimensional universal word vectors:

$$\Upsilon = \lambda\sqrt{v}\mathbf{V}_{vk} \quad (1)$$

where v is the size of vocabulary, $\mathbf{V}$ is the matrix of right singular vectors, λ is the scaling factor that controls the variance of the data.

The word vectors are extracted from the training part of the UD corpora for all words whose frequencies exceed 5, resulting in $204,024$ unique words. The number of dimensions, k, is set to 50 and the scaling parameter λ is set to 0.1 as suggested by Basirat and Nivre (2017). Adding these pre-trained word embeddings improved results on development sets by 0.44 points on average.

5 Multilingual Models

The shared task contained four surprise languages, Buryat, Kurmanji, North Sami, and Upper Sorbian, for which there was no data available until the last week, when we had a few sample sentences for each language. Two of the ordinary languages, Kazakh and Uyghur, had a similar situation, since they had less than 50 sentences in their training data. We therefore decided to treat those two languages like the surprise languages.

For segmentation, we utilized the small amount of available annotated data as development sets. We applied all the segmentation models trained on larger treebanks and adopted the one that achieved the highest F1-score as the segmentation model for the surprise language. We thus selected Bulgarian for Buryat, Slovenian for North Sami, Czech for Upper Sorbian, Turkish for Kurmanji, Russian for Kazakh as well as Persian for Uyghur.

For parsing, we trained our parser on a small set of languages. For each surprise language, we used the little data we had for that language, and in addition a set of other languages, which we will call support languages. In this setting we took advantage of the language embedding implemented in the parser. Since the treebanks for the support languages have very different sizes, we limited the number of sentences from each treebank used per epoch to 2263 for North Sami and 2500 for the other languages, in order ot use a more balanced sample. For each epoch we randomly picked a new sample of sentences for each treebank larger than this ceiling. We chose the support languages for each surprise language based on four criteria:

- Language relatedness, by including the languages that were most closely related to each surprise language.
- Script, by choosing at least one language sharing the same script as each surprise language, which might help our character embeddings.
- Geographical closeness to the surprise language, since geographically close languages often influence each other and can share many traits and have loan words.
- Performance of single models, by evaluating individual models for all other languages on each surprise language, and choosing support languages from the set of best performing languages.

We used a single multi-lingual model for Kazakh and Uyghur, since they are related. Table 3 shows the support languages used for each surprise language. Since we used all available surprise language data in the training, we could not use it also as development data, to pick the best epoch. We instead used the average LAS score on the development data for all support languages that had available development data. We did not use the pseudo-projective method for the surprise languages, and we did not use pre-trained word embeddings.

6 Results and Discussion

Table 4 summarizes the results for the Uppsala system with respect to dependency accuracy (LAS

Surprise	Support languages
Buryat	Russian-SynTagRus^{gps}, Russian^{gs}, Japanese^{pr}, Kazakh^{ps}, Bulgarian^{s}
Kurmanji	Turkish^{gs}, Persian^{r}, Finnish-FTB^{ps}, German^{ps}, Slovenian-SST^{ps}
North Sami	Finnish^{rs}, Finnish-FTB^{prs}, Estonian^{rs}, Hungarian^{prs}, Norwegian-Nynorsk^{gps}
Upper Sorbian	Czech^{prs}, Slovak^{prs}, Slovenian^{prs}, Polish^{prs}, German^{gs}
Kazakh+Uyghur	Russian-SynTagRus^{gps}, Hungarian^{p}, Turkish^{pr}, Persian^{s}, Arabic^{s}

Table 3: Support languages, and treebanks, used for each surprise language. Superscripts show reason for inclusion: r(elated), s(cript), g(eography), p(erformance).

F1) as well as sentence and word segmentation. For each metric, we report our official test score (Test), the corrected score after eliminating the two bugs described in Section 2 and Section 3 (Corr),[1] and the difference between the corrected score and the official UDPipe baseline (Straka et al., 2016) (positive if we beat the baseline and negative otherwise). To make the table somewhat more readable, we have also added a simple color coding. Scores that are significantly higher/lower than the baseline are marked with two shades of green/red, with brighter colors for larger differences. Thresholds have been set to 1 and 3 points for LAS, 0.5 and 1 points for Sentences, and 0.1 and 0.5 points for Words.

Looking first at the LAS scores, we see that our system improves over the baseline in most cases and by a comfortable margin. In addition, we think we can distinguish three clear patterns:

- Our results are substantially worse than the baseline (only) on the six low-resource languages. This indicates that our cross-lingual models perform poorly without the help of part-of-speech tags when it has little training data. It should, however, also be kept in mind that the baseline had a special advantage here as it was allowed to train segmenters and taggers using jack-knifing on the test sets.

- Our results are substantially better than the baseline on languages with writing systems that differ (more or less) from European style alphabetic scripts, including Arabic, Chinese, Hebrew, Japanese, Korean, and Vietnamese. For all languages except Korean, this can be partly (but not wholly) explained by more accurate word segmentation results.

- Our results are substantially better than the baseline for a number of morphologically rich languages, including Ancient Greek, Arabic, Basque, Czech, Finnish, German, Latin, Polish, Russian, and Slovenian. This shows that character-based representations are effective in capturing morphological regularities and compensate for the lack of explicit morphological features.

To further investigate the efficiency of our cross-lingual models, we ran them for two of the support languages with medium size training data that were not affected by the capping of data. Table 5 shows the results of this investigation. For Estonian the North Sami cross-lingual model that includes the closely related Finnish, was better than the monolingual model. For Hungarian, on the other hand, the monolingual model was better than both cross-lingual models. The model for North Sami, with related languages did perform better than the model for Kazakh+Uyghur with only unrelated languages, however. These results indicate that cross-lingual training without part-of-speech tags can help for a language with a medium sized treebank, but it seems that closely related support languages are needed, which was not the case for any of the surprise languages.

For word segmentation, we have already noted that our universal model works well on some of the most challenging languages, such as Chinese, Japanese and Vietnamese, and also on the Semitic languages Arabic and Hebrew. This is not surprising, given that the model was first developed for Chinese word segmentation, but it is interesting to see that it generalizes well and gives competitive results also on European style alphabetic scripts, where it is mostly above or very close to the baseline. After fixing the bug mentioned in Section 2, our word segmentation results are in fact second best overall, only 0.02 below the best system.

The sentence segmentation results are generally harder to interpret, with much greater variance and

[1] Note that the overview paper mentions the second of these bugs (i.e. the dynet bug) and reports our results with only that bug fixed.

	LAS F1			Sentences			Words		
Language	**Test**	**Corr**	**Diff**	**Test**	**Corr**	**Diff**	**Test**	**Corr**	**Diff**
ar	65.96	68.68	3.38	77.32	78.21	-6.36	94.81	94.99	1.30
ar_pud	47.34	50.70	7.56	97.18	98.66	-1.34	94.32	95.30	4.48
bg	81.25	85.38	1.74	93.36	95.23	2.40	99.70	99.91	0.00
bxr	17.14	18.01	-13.49	86.93	87.37	-4.44	97.77	97.71	-1.64
ca	85.42	87.08	1.69	99.43	99.59	0.64	99.78	99.79	-0.18
cs	85.88	86.83	3.96	93.97	92.79	0.76	99.96	99.98	0.08
cs_cac	83.66	85.75	3.29	99.76	99.68	-0.32	99.97	99.99	0.00
cs_cltt	59.84	75.67	4.03	92.99	96.95	1.89	99.54	99.78	0.43
cs_pud	80.21	82.27	2.47	94.18	95.55	-0.88	98.42	99.25	-0.04
cu	57.88	67.04	4.28	39.71	43.72	7.67	99.73	99.94	-0.02
da	70.63	77.70	4.32	81.12	83.41	4.05	99.93	100.00	0.31
de	72.61	75.27	6.16	80.47	81.47	2.36	99.44	99.67	0.02
de_pud	68.04	70.90	4.37	87.16	86.83	0.34	96.42	96.43	-1.57
el	72.77	80.46	1.20	90.38	91.09	0.30	99.83	99.80	-0.08
en	75.88	79.62	3.78	76.91	80.26	7.04	98.38	99.05	0.38
en_lines	67.52	75.80	2.86	86.84	87.17	1.33	99.82	99.96	0.02
en_partut	63.55	76.11	2.47	98.20	98.10	0.59	99.55	99.54	0.05
en_pud	75.61	80.49	1.54	95.28	96.15	-0.98	99.45	99.59	-0.07
es	82.17	84.26	2.79	95.37	94.16	0.01	99.81	99.84	0.15
es_ancora	84.60	86.79	3.01	98.06	98.46	1.41	99.89	99.92	-0.03
es_pud	78.16	79.01	1.36	93.41	93.39	-0.03	99.39	99.34	-0.13
et	49.01	58.67	-0.12	92.74	93.23	8.03	99.69	99.90	0.13
eu	69.84	73.82	4.67	99.67	100.00	0.42	99.97	100.00	0.04
fa	76.13	81.89	2.65	98.75	99.50	1.50	99.32	99.61	-0.03
fi	74.59	78.41	4.66	90.88	91.48	6.92	99.62	99.71	0.08
fi_ftb	71.85	76.25	2.22	86.98	87.16	3.33	99.91	99.99	0.11
fi_pud	76.22	80.05	1.40	92.02	91.64	-2.03	99.39	99.59	-0.02
fr	80.36	83.66	2.91	93.85	94.32	0.73	99.50	99.53	0.66
fr_partut	69.17	80.84	3.46	99.13	99.50	1.50	99.01	99.50	0.55
fr_pud	73.51	75.25	1.62	93.52	91.33	-0.99	97.38	97.34	-0.83
fr_sequoia	74.96	82.85	2.87	81.89	84.95	1.20	99.31	99.48	0.42
ga	52.81	63.35	1.83	95.70	95.35	-0.46	99.62	99.78	0.49
gl	74.09	79.01	1.70	96.36	96.83	0.68	99.91	99.96	0.04
gl_treegal	56.79	65.85	0.03	82.71	83.79	2.16	98.42	98.23	-0.39
got	56.69	62.62	2.81	29.65	35.01	7.16	100.00	100.00	0.00
grc	50.94	58.83	2.79	98.70	98.93	0.50	96.78	99.98	0.03
grc_proiel	63.86	69.04	3.82	49.31	48.86	5.75	99.99	99.98	-0.02
he	63.72	67.75	10.52	99.29	99.69	0.30	91.18	91.19	6.37
hi	74.34	89.13	2.36	99.29	99.11	-0.09	92.74	99.99	-0.01
hi_pud	45.15	53.31	2.46	94.85	95.00	4.17	92.27	98.65	0.84
hr	75.43	79.51	2.33	97.75	97.25	0.33	99.90	99.91	-0.02
hsb	45.63	47.92	-5.91	91.65	89.88	-0.81	99.28	98.76	-1.08
hu	54.55	65.90	1.60	96.56	97.65	3.80	99.85	99.89	0.07
id	72.11	76.13	1.52	92.66	93.55	2.40	100.00	100.00	0.01
it	84.84	87.33	2.05	99.07	99.38	2.28	99.85	99.86	0.13
it_pud	83.28	85.59	1.89	93.39	93.90	-2.68	99.27	99.28	0.11
ja	65.71	81.54	9.33	94.92	94.92	0.00	84.26	93.59	3.91
ja_pud	71.80	83.26	6.98	97.31	97.31	2.42	86.34	94.30	3.24
kk	18.24	17.14	-7.37	87.52	86.26	4.88	96.56	96.46	1.55
kmr	19.37	20.39	-11.96	94.49	94.08	-2.94	97.15	97.06	-1.79
ko	69.87	74.72	15.63	92.39	93.01	-0.04	99.63	99.99	0.26
la	38.93	46.26	2.49	98.04	97.41	-0.68	100.00	100.00	0.01
la_ittb	80.04	82.34	5.36	94.34	92.93	-0.31	99.97	99.99	0.00
la_proiel	58.74	63.17	5.63	30.24	34.66	8.86	99.99	100.00	0.00
lv	52.36	59.75	-0.20	93.45	93.65	-4.94	99.20	99.13	0.22
nl	69.83	74.41	5.51	75.15	76.16	-0.98	99.73	99.85	-0.03
nl_lassysmall	77.56	83.58	5.43	85.33	87.00	8.38	99.85	99.97	0.04
no_bokmaal	83.22	86.04	2.77	96.44	96.20	0.44	99.84	99.87	0.12
no_nynorsk	81.12	84.41	2.85	94.56	93.67	2.44	99.93	99.92	0.07
pl	77.39	82.33	3.55	98.91	99.46	0.55	99.90	99.93	0.05
pt	80.97	83.25	1.14	90.33	90.43	0.64	99.37	99.45	-0.07
pt_br	86.15	88.19	2.83	96.51	97.04	0.20	99.80	99.87	0.03
pt_pud	72.43	74.48	0.52	93.58	94.50	-1.15	98.39	98.48	-0.94
ro	79.40	81.68	1.80	96.57	96.02	2.60	99.77	99.75	0.11
ru	71.65	77.99	3.96	97.16	96.91	0.49	99.83	99.90	-0.01
ru_pud	65.22	70.78	2.47	98.66	98.80	-0.15	97.31	97.34	0.16
ru_syntagrus	88.04	89.61	2.85	98.64	98.78	0.97	99.51	99.63	0.06
sk	69.35	75.98	3.23	85.32	87.17	3.64	99.97	99.96	-0.04
sl	80.14	84.16	3.01	98.67	98.11	-1.13	99.96	99.97	0.01
sl_sst	36.97	46.76	0.31	19.03	19.52	2.80	97.75	100.00	0.18
sme	11.70	11.72	-18.88	98.27	97.59	-1.20	98.44	96.75	-3.13
sv	73.45	79.86	3.13	97.26	95.96	-0.41	99.86	99.77	-0.07
sv_lines	69.42	76.37	2.08	87.89	88.12	1.68	99.86	99.99	0.01
sv_pud	62.40	69.52	-1.10	84.63	81.14	-9.06	98.56	98.47	0.21
tr	48.29	52.84	-0.35	96.29	96.44	-0.19	96.57	97.51	-0.38
tr_pud	29.79	32.84	-1.69	92.08	90.75	-3.16	96.82	96.93	0.31
ug	28.35	30.98	-3.20	68.76	69.36	5.81	97.82	98.74	0.22
uk	47.00	59.33	-1.43	90.04	92.18	-0.41	99.41	99.52	-0.29
ur	64.96	79.31	2.62	98.60	98.60	0.28	94.55	100.00	0.00
vi	37.99	42.68	5.21	87.30	89.49	-3.10	86.63	86.70	4.23
zh	60.47	65.25	7.85	98.20	98.80	0.61	93.81	93.43	4.52
Average	65.11	70.49	2.14	89.03	89.48	0.99	98.20	98.79	0.30

Table 4: Results for LAS F1, sentence and word segmentation. Test = official test score; Corr = corrected score; Diff = difference Corr − Baseline.

Language	Models		
	sme	kk-ug	mono
Hungarian	62.67	61.64	65.91
Estonian	59.59	–	58.46

Table 5: LAS F1 scores comparing cross-lingual and monolingual models.

really low scores especially for some of the classical languages that lack modern punctuation. Nevertheless, we can conclude that performing sentence segmentation jointly with word segmentation is a viable approach, as our system achieved the second highest score of all systems on sentence segmentation in the official test results. After bug fixing, it is the best of all.

All in all, we are pleased to see that a bare-bones model, which does not make use of part-of-speech tags, morphological features or lemmas, can give reasonable performance on a wide range of languages. At the time of writing, our corrected test results put us in the top ten on the list of post-evaluation results for LAS F1, with some of the best scores for word and sentence segmentation.

7 Conclusion

We have described the Uppsala submission to the CoNLL 2017 shared task on parsing from raw text to universal dependencies. The system consists of a segmenter, which extracts words and sentences from a raw text, and a parser, which builds a dependency tree over the words of each sentence, without relying on part-of-speech tags or any other explicit morphological analysis. Our parsing results (after correcting two bugs) are on average 2.14 points above the baseline, despite very poor performance on surprise languages, and the system has competitive results especially on languages with rich morphology and/or non-European writing systems. Given the simplicity of our system, we find the results very encouraging.

There are many different lines of future research that we want to pursue. First of all, we want to explore the use of multilingual models with language embeddings, trained on much larger data sets than was practically possible for the shared task. In this context, we also want to investigate the effectiveness of our multilingual word embeddings based on universal part-of-speech tags, deriving them from large parsed corpora instead of the small training sets that were used for the shared task. Finally, we want to extend the parser so that it can jointly predict part-of-speech tags and (selected) morphological features. This will allow us to systematically study the effect of using explicit linguistic categories, as opposed to just relying on inference from raw words and characters. For segmentation, we want to investigate how our model deals with multiword tokens across languages.

Acknowledgments

We are grateful to the shared task organizers and to Dan Zeman in particular, and we acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu). Joakim Nivre's contributions to this work were supported by grant 2016-01817 of the Swedish Research Council. Our parser will be made available in the NLPL dependency parsing laboratory.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics* 4:431–444.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 349–359.

Ali Basirat and Joakim Nivre. 2017. Real-valued syntactic word vectors (RSV) for greedy neural dependency parsing. In *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*. pages 21–28.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics* 1:415–428.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*. pages 149–164.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*. pages 249–256.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics* 1:403–414.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 1045–1053.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-first dependency parsing with hierarchical tree LSTMs. *Transactions of the Association for Computational Linguistics* 4:445–461.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

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 (ACL)*. pages 673–682.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 1064–1074.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017a. 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.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017b. 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, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*. pages 915–932.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*. pages 2216–2219.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 99–106.

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.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF. ArXiv e-prints: 1704.01314 (cs.CL).

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 (LREC 2016)*. European Language Resources Association, Portorož, Slovenia.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation* 26(2):289–315.

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.