A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages

Clara Vania1  Yova Kementchedjhieva2  Anders Søgaard2  Adam Lopez1
1School of Informatics, University of Edinburgh, UK
2University of Copenhagen, Copenhagen, Denmark
c.vania@ed.ac.uk, {yova|soegaard}@di.ku.dk, alopez@inf.ed.ac.uk

Abstract

Parsers are available for only a handful of the world’s languages, since they require lots of training data. How far can we get with just a small amount of training data? We systematically compare a set of simple strategies for improving low-resource parsers: data augmentation, which has not been tested before; cross-lingual training; and transliteration. Experimenting on three typologically diverse low-resource languages—North Sámi, Galician, and Kazah—We find that (1) when only the low-resource treebank is available, data augmentation is very helpful; (2) when a related high-resource treebank is available, cross-lingual training is helpful and complements data augmentation; and (3) when the high-resource treebank uses a different writing system, transliteration into a shared orthographic spaces is also very helpful.

1 Introduction

Large annotated treebanks are available for only a tiny fraction of the world’s languages, and there is a wealth of literature on strategies for parsing with few resources (Hwa et al., 2005; Zeman and Resnik, 2008; McDonald et al., 2011; Søgaard, 2011). A popular approach is to train a parser on a related high-resource language and adapt it to the low-resource language. This approach benefits from the availability of Universal Dependencies (UD; Nivre et al., 2016), prompting substantial research (Tiedemann and Agić, 2016; Agić, 2017; Rosa and Mareček, 2018), along with the VarDial and the CoNLL UD shared tasks (Zampieri et al., 2017; Zeman et al., 2017, 2018).

But low-resource parsing is still difficult. The organizers of the CoNLL 2018 UD shared task (Zeman et al., 2018) report that, in general, results on the task’s nine low-resource treebanks “are extremely low and the outputs are hardly useful for downstream applications.” So if we want to build a parser in a language with few resources, what can we do? To answer this question, we systematically compare several practical strategies for low-resource parsing, asking:

1. What can we do with only a very small target treebank for a low-resource language?
2. What can we do if we also have a source treebank for a related high-resource language?
3. What if the source and target treebanks do not share a writing system?

Each of these scenarios requires different approaches. Data augmentation is applicable in all scenarios, and has proven useful for low-resource NLP in general (Fadaee et al., 2017; Bergmanis et al., 2017; Sahin and Steedman, 2018). Transfer learning via cross-lingual training is applicable in scenarios 2 and 3. Finally, transliteration may be useful in scenario 3.

To keep our scenarios as realistic as possible, we assume that no taggers are available since this would entail substantial annotation. Therefore, our neural parsing models must learn to parse from words or characters—that is, they must be lexicalized—even though there may be little shared vocabulary between source and target treebanks. While this may intuitively seem to make cross-lingual training difficult, recent results have shown that lexical parameter sharing on characters and words can in fact improve cross-lingual parsing (de Lhoneux et al., 2018); and that in some circumstances, a lexicalized parser can outperform a delexicalized one, even in a low-resource setting (Falenska and Çetinoğlu, 2017).

We experiment on three language pairs from different language families, in which the first of each is a genuinely low-resource language: North
Sámi and Finnish (Uralic); Galician and Portuguese (Romance); and Kazakh and Turkish (Turkic), which have different writing systems\(^1\). To avoid optimistic evaluation, we extensively experiment only with North Sámi, which we also analyse to understand why our cross-lingual training outperforms the other parsing strategies. We treat Galician and Kazakh as truly held-out, and test only our best methods on these languages. Our results show that:

1. When no source treebank is available, data augmentation is very helpful: dependency tree morphing improves labeled attachment score (LAS) by as much as 9.3%. Our analysis suggests that syntactic rather than lexical variation is most useful for data augmentation.

2. When a source treebank is available, cross-lingual parsing improves LAS up to 16.2%, but data augmentation still helps, by an additional 2.6%. Our analysis suggests that improvements from cross-lingual parsing occur because the parser learns syntactic regularities about word order, since it does not have access to POS and has little reusable information about word forms.

3. If source and target treebanks have different writing systems, transliterating them to a common orthography is very effective.

2 Methods

We describe three techniques for improving low-resource parsing: (1) two data augmentation methods which have not been applied before for dependency parsing, (2) cross-lingual training, and (3) transliteration.

2.1 Data augmentation by dependency tree morphing (Morph)

Sahin and Steedman (2018) introduce two operations to augment a dataset for low-resource POS tagging. Their method assumes access to a dependency tree, but they do not test it for dependency parsing, which we do here for the first time. The first operation, cropping, removes some parts of a sentence to create a smaller or simpler, meaningful sentence. The second operation, rotation, keeps all the words in the sentence but re-orders subtrees attached to the root verb, in particular those attached by NSUBJ (nominal subject), OBJ (direct object), IOBJ (indirect object), or OBL (oblique nominal) dependencies. Figure 1 illustrates both operations.

It is important to note that while both operations change the set of words or the word order, they do not change the dependencies. The sentences themselves may be awkward or ill-formed, but the corresponding analyses are still likely to be correct, and thus beneficial for learning. This is because they provide the model with more examples of variations in argument structure (cropping) and in constituent order (rotation), which may benefit languages with flexible word order and rich morphology. Some of our low-resource languages have these properties—while North Sámi has a fixed word order (SVO), Galician and Kazakh have relatively free word order. All three languages use case marking on nouns, so word order may not be as important for correct attachment.

Both rotation and cropping can produce many trees. We use the default parameters given in (Sahin and Steedman, 2018).

---

\(^1\)We select high-resource language based on language family, since it is the most straightforward way to define language relatedness. However, other measurement (e.g., WALS (Dryer and Haspelmath, 2013) properties) might be used.
2.2 Data augmentation by nonce sentence generation (Nonce)

Our next data augmentation method is adapted from Gulordava et al. (2018). The main idea is to create nonce sentences by replacing some of the words which have the same syntactic annotations. For each training sentence, we replace each content word—nouns, verbs, or adjective—with an alternative word having the same universal POS, morphological features, and dependency label. Specifically, for each content word, we first stochastically choose whether to replace it; then, if we have chosen to replace it, we uniformly sample the replacement word type meeting the corresponding constraints. For instance, given a sentence “He borrowed a book from the library,” we can generate the following sentences:

a. He bought a book from the shop.

b. He wore a umbrella from the library.

This generation method is only based on syntactic features (i.e., morphology and dependency labels), so it sometimes produces nonsensical sentences like 1b. But since we only replace words if they have the same morphological features and dependency label, this method preserves the original tree structures in the treebank. Following (Gulordava et al., 2018), we generate five nonce sentences for each original sentence.

2.3 Cross-lingual training

When a source treebank is available, model transfer is a viable option. We perform model transfer by cross-lingual parser training: we first train on both source and target treebanks to produce a single model, and then fine tune the model only on the target treebank. In our preliminary experiments (Appendix A), we found that fine tuning on the target treebank was effective in all settings, so we use it in all applicable experiments reported in this paper.

2.4 Transliteration

Two related languages might not share a writing system even when they belong to the same family. We evaluate whether a simple transliteration would be helpful for cross-lingual training in this case. In our study, the Turkish treebank is written in extended Latin while the Kazakh treebank is written in Cyrillic. This difference potentially makes model transfer less useful, and means we might not be able to leverage lexical similarities between the two languages. We pre-process both treebanks by transliterating them to the same “pivot” alphabet, basic Latin.3

The mapping from Turkish is straightforward. Its alphabet consists of 29 letters, 23 of which are in basic Latin. The other six letters, ‘ç’, ‘ğ’, ‘ı’, ‘ö’, ‘ş’, and ‘ü’, add diacritics to basic Latin characters, facilitating different pronunciations.4 We map these to their basic Latin counterparts, e.g., ‘ç’ to ‘c’. For Kazakh, we use a simple dictionary created by a Kazakh computational linguist to map each Cyrillic letter to the basic Latin alphabet.5

3 Experimental Setup

3.1 Dependency Parsing Model

We use the Uppsala parser, a transition-based neural dependency parser (de Lhoneux et al., 2017a,b; Kiperwasser and Goldberg, 2016). The parser uses an arc-hybrid transition system (Kuhlmann et al., 2011), extended with a static-dynamic oracle and SWAP transition to allow non-projective dependency trees (Nivre, 2009).

Let $w = w_0, \ldots, w_{|w|}$ be an input sentence of length $|w|$ and let $w_0$ represent an artificial ROOT token. We create a vector representation for each input token $w_i$ by concatenating (:) its word embedding, $e_w(w_i)$ and its character-based word embedding, $e_c(w_i)$:

$$x_i = [e_w(w_i); e_c(w_i)] \quad (1)$$

Here, $e_c(w_i)$ is the output of a character-level bidirectional LSTM (biLSTM) encoder run over the characters of $w_i$ (Ling et al., 2015); this makes the model fully open-vocabulary, since it can produce representations for any character sequence. We then obtain a context-sensitive encoding $h_i$ using a word-level biLSTM encoder:

$$h_i = [\text{LSTM}_f(x_{i+1}); \text{LSTM}_b(x_{i-1})] \quad (2)$$

We then create a configuration by concatenating the encoding of a fixed number of words on the top of the stack and the beginning of the buffer. Given

3Another possible pivot is phonemes (Tsvetkov et al., 2016). We leave this as future work.
4https://www.omniglot.com/writing/turkish.htm
5The mapping from Kazakh Cyrillic into basic Latin alphabet is provided in Appendix B
Table 1: Train/dev split used for each treebank.

<table>
<thead>
<tr>
<th>Language</th>
<th>Treebank ID</th>
<th>train</th>
<th>dev.</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finnish</td>
<td>fi_tdt</td>
<td>14981</td>
<td>1875</td>
<td>1555</td>
</tr>
<tr>
<td>North Sámi</td>
<td>sme_giella</td>
<td>1128</td>
<td>1129</td>
<td>865</td>
</tr>
<tr>
<td>Portuguese</td>
<td>pt_bosque</td>
<td>8329</td>
<td>560</td>
<td>477</td>
</tr>
<tr>
<td>Galician</td>
<td>gl_treegal</td>
<td>300</td>
<td>300</td>
<td>400</td>
</tr>
<tr>
<td>Turkish</td>
<td>tr_imst</td>
<td>3685</td>
<td>975</td>
<td>975</td>
</tr>
<tr>
<td>Kazakh</td>
<td>kk_ktb</td>
<td>15</td>
<td>16</td>
<td>1047</td>
</tr>
</tbody>
</table>

To train cross-lingual models, we use the strategy of de Lhoneux et al. (2018) for parameter sharing, which uses soft sharing for word and character parameters, and hard sharing for the MLP parameters. Soft parameter sharing uses a language embedding, which, in theory, learns what parameters to share between the two languages. Let $c_j$ be an embedding of character $c_j$ in a token $w_i$ from the treebank of language $k$, and let $l_k$ be the language embedding. For sharing on characters, we concatenate character and language embedding: $[c_j; l_k]$ for input to the character-level biLSTM. Similarly, for input to the word-level biLSTM, we concatenate the language embedding to the word embedding, modifying Eq. 1 to

$$x_i = [e_w(w_i); e_c(w_i); l_k]$$  

We use Universal Dependencies (UD) treebanks version 2.2 (Nivre et al., 2018). None of our target treebanks have a development set, so we generate new train/dev splits by 50:50 (Table 1). Having large development sets allow us to perform better analysis for this study.

3.3 Datasets

We use Universal Dependencies (UD) treebanks version 2.2 (Nivre et al., 2018). None of our target treebanks have a development set, so we generate new train/dev splits by 50:50 (Table 1). Having large development sets allow us to perform better analysis for this study.

4 Parsing North Sámi

North Sámi is our largest low-resource treebank, so we use it for a full evaluation and analysis of different strategies before testing on the other languages. To understand the effect of target treebank size, we generate three datasets with different training sizes: $T_{10}$ (~10%), $T_{50}$ (~50%), and $T_{100}$ (100%). Table 2 reports the number of training sentences after we augment the data using methods described in Section 2. We apply MORPH and NONCE separately to understand the effect of each method and to control the amount of noise in the augmented data.

We employ two baselines: a monolingual model (§3.1) and a cross-lingual model (§2.3), both without data augmentation. The monolingual model acts as a simple baseline, to resemble a situation when the target treebank does not have any source treebank (i.e., no available treebanks from related languages). The cross-lingual model serves as a strong baseline, simulating a case when there is a source treebank. We compare both baselines to models trained with MORPH and NONCE augmentation methods. Table 3 reports our results, and we review our motivating scenarios below.

Scenario 1: we only have a very small target treebank. In the monolingual experiments, we observe that both dependency tree morphing (MORPH) and nonce sentence generation (NONCE) improve performance, indicating the strong benefits of data augmentation when there are no other resources available except the target treebank itself. In particular, when the number of training data is the lowest ($T_{10}$), data augmentation improves performance up to 9.3% LAS.

Scenario 2: a source treebank is available. We see that the cross-lingual training (cross-base) performs better than monolingual models even with augmentation. For the $T_{10}$ setting, cross-base achieves almost twice as much as the monolingual baseline (mono-base). The benefits of data augmentation are less evident in the cross-lingual setting, but in the $T_{10}$ scenario, data augmentation still clearly helps. Overall, cross-lingual combined with data augmentation yields the best result.
### 4.1 What is learned from Finnish?

Why do cross-lingual training and data augmentation help? To put this question in context, we first consider their relationship. Finnish and North Sámi are mutually unintelligible, but they are typologically similar: of the 49 (mostly syntactic) linguistic features annotated for North Sámi in the Word Atlas of Languages (WALS; Dryer and Haspelmath, 2013), Finnish shares the same values for 42 of them.\(^6\) Despite this and their phylogenetic and geographical relatedness, they share very little vocabulary: only 6.5% of North Sámi tokens appear in Finnish data, and these words are either proper nouns or closed class words such as pronouns or conjunctions. However, both languages do share many character-trigrams (72.5%, token-level), especially on suffixes.

Now we turn to an analysis of the \(\mathcal{T}_{10}\) data setting, where we see the largest gains for all methods.

### 4.2 Analysis of data augmentation

For dependency parsing, POS features are important because they can provide strong signals whether there exists dependency between two words in a given sentence. For example, *subject* and *object* dependencies often occur between a *NOUN* and a *VERB*, as can be seen in Fig. 1a. We investigate the extent to which data augmentation is useful for learning POS features, using diagnostic classifiers (Veldhoen et al., 2016; Adi et al., 2016; Shi et al., 2016) to probe our model representations. Our central question is: do the models learn useful representations of POS, despite having no direct access to it? And if so, is this helped by data augmentation?

After training each model, we freeze the parameters and generate *context-dependent* representations (i.e., the output of *word-level* biLSTM, \(h\) in Eq. 2), for the training and development data. We then train a feed-forward neural network classifier to predict the POS tag of each word, using only the representation as input. To filter out the effect of cross-lingual training, we only analyze representations trained using the *monolingual* models. Our training and development data consists of 6321 and 7710 tokens, respectively. The percentage of OOV tokens is 40.5%.

Table 4 reports the POS prediction accuracy. We observe that representations generated with monolingual MORPH seem to learn better POS, for most of the tags. On the other hand, representations generated with monolingual NONCE sometimes produce lower accuracy on some tags; only on nouns the accuracy is better than monolingual MORPH. We hypothesize that this is because NONCE sometimes generates meaningless sentences which confuse the model. In parsing this effect is less apparent, mainly because monolingual NONCE has the poorest POS representation for infrequent tags (%dev), and better representa-

\begin{table}[h]
\centering
\begin{tabular}{lcccccc}
\hline
POS & \%dev & baseline & \%diff. with & \%diff. with \\
 & & & +Morph & +Nonce & +Morph & +Nonce \\
\hline
INTJ & 0.1 & 0.0 & 20.0 & 20.0 \\
PART & 1.5 & 70.1 & 7.7 & 0.8 \\
NUM & 1.9 & 19.2 & 15.1 & -4.1 \\
ADP & 1.9 & 15.7 & 24.5 & 19.7 \\
SCONJ & 2.4 & 57.8 & 5.9 & 7.6 \\
AUX & 3.2 & 26.3 & 27.2 & -4.9 \\
CCONJ & 3.4 & 91.3 & -0.8 & -4.2 \\
PROPN & 4.7 & 5.9 & 5.9 & -5.9 \\
ADJ & 6.5 & 12.7 & 3.8 & 0.2 \\
ADV & 9.0 & 0.0 & 1.18 & 11.5 \\
PRON & 13.4 & 63.2 & 5.4 & -2.7 \\
VERB & 25.7 & 72.4 & -6.2 & -4.5 \\
NOUN & 26.4 & 67.0 & 8.6 & 13.2 \\
\hline
\end{tabular}
\caption{Results for the monolingual POS predictions, ordered by the frequency of each tag in the dev split (%dev). %diff shows the difference between each augmentation method and monolingual models.}
\end{table}
Table 5: Most similar Finnish words for each North Sámi word based on cosine similarity.

<table>
<thead>
<tr>
<th>North Sámi</th>
<th>char-level</th>
<th>word-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>børrat (VERB; eat)</td>
<td>herrat (NOUN; gentleman)</td>
<td>käydä (VERB; go)</td>
</tr>
<tr>
<td>kerrat (NOUN; time)</td>
<td>otan (VERB; take)</td>
<td></td>
</tr>
<tr>
<td>naurat (VERB; laugh)</td>
<td>sain (VERB; get)</td>
<td></td>
</tr>
<tr>
<td>veahki (NOUN; help)</td>
<td>nuhki (VERB; sniff)</td>
<td>tyhjäksi (ADJ; empty)</td>
</tr>
<tr>
<td>vääki (NOUN; power)</td>
<td>avärsi (VERB; expand)</td>
<td>johonki (PRON; something)</td>
</tr>
<tr>
<td>diivras (ADJ; expensive)</td>
<td>harras (ADJ; devout)</td>
<td>välittömääntä (ADJ; essential)</td>
</tr>
<tr>
<td>reipas (ADJ; brave)</td>
<td>sarjaporras (NOUN; series)</td>
<td>kiippulukkyysenpi (ADJ; competitive)</td>
</tr>
</tbody>
</table>

Table 6: # of North Sámi tokens for which the most similar Finnish word has the same POS.

<table>
<thead>
<tr>
<th>POS</th>
<th>char-level (%)</th>
<th>word-level (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>12.1</td>
<td>37.1</td>
</tr>
<tr>
<td>NOUN</td>
<td>55.8</td>
<td>63.5</td>
</tr>
<tr>
<td>PRON</td>
<td>12.9</td>
<td>68.0</td>
</tr>
<tr>
<td>VERB</td>
<td>34.2</td>
<td>69.0</td>
</tr>
</tbody>
</table>
highlight confusions that are more common for the cross-lingual model, while red cells highlight those more common for the monolingual model. For head prediction (last column), we observe that monolingual model makes higher errors especially for nominals and modifier words. In cases when both both models predict the correct heads, we observe that cross-lingual training gives further improvements in predicting most of the labels. In particular, regarding the “rootless” trees discussed before, we see evidence that cross-lingual training helps in predicting the correct root index, and the correct root label.

5 Parsing truly low-resource languages

Now we turn to two truly low-resource treebanks: Galician and Kazakh. These treebanks are most analogous to the North Sami $T_{10}$ setting and therefore we apply the best approach, cross-lingual training with MORPH augmentation. Table 1 provides the statistics of the augmented data. For Galician, we use the Portuguese treebank as source while for Kazakh we use Turkish. Portuguese and Galician have high vocabulary overlap; 62.9% of Galician tokens appear in Portuguese data, while for Turkish and Kazakh they do not share vocabulary since they use different writing systems. However, after transliterating them into the same basic Latin alphabet, we observe that 9.5% of Kazakh tokens appear in the Turkish data. Both language pairs also share many (token-level) character trigrams: 96.0% for Galician-Portuguese and 66.3% for transliterated Kazakh-Turkish.

To compare our best approach, we create two baselines: (1) a pre-trained parsing model of the source treebank (zero-shot learning), and (2) a cross-lingual model initialized with monolingual pre-trained word embeddings. The first serves as a weak baseline, in a case where training on the target treebank is not possible (e.g., Kazakh only has 15 sentences for training). The latter serves as

Table 7: LAS results on development sets. zero-shot denotes results where we predict using model trained only on the source treebank.
a strong baseline, in a case when we have access to pre-trained word embeddings, for the source and/or the target languages.

We treat a pre-trained word embedding as an external embedding, and concatenate it with the other representations, i.e., modifying Eq. 3 to \( x_i = [e_w(w_i); e_p(w_i); e_c(w_i); l_k] \), where \( e_p(w_i) \) represents a pre-trained word embedding of \( w_i \), which we update during training. We use the pre-trained monolingual fastText embeddings (Bojanowski et al., 2017).\(^9\) We concatenate the source and target pre-trained word embeddings.\(^10\) For our experiments with transliteration (§2.4), we transliterate the entries of both the source and the target pre-trained word embeddings.

### 5.1 Experimental results

Table 7 reports the LAS performance on the development sets. MORPH augmentation improves performance over the zero-shot baseline and achieves comparable or better LAS with a cross-lingual model trained with pre-trained word embeddings.

Next, we look at the effects of transliteration (see Kazakh vs Kazakh (translit.) in Table 7). In the zero-shot experiments, simply mapping both Turkish and Kazakh characters to the Latin alphabet improves accuracy from 12.5 to 21.2 LAS. Cross-lingual training with MORPH further improves performance to 36.7 LAS.

### 5.2 Comparison with CoNLL 2018

To see how our best approach (i.e., cross-lingual model with MORPH augmentation) compares with the current state-of-the-art models, we compare it to the recent results from CoNLL 2018 shared task. Training state-of-the-art models may require lots of engineering and data resources. Our goal, however, is not to achieve the best performance, but rather to systematically investigate how far simple approaches can take us. We report performance of the following: (1) the shared task baseline model (UDPipe v1.2; Straka and Straková, 2017) and (2) the best system for each treebank, (3) our best approach, and (4) a cross-lingual model with fastText embeddings.

Table 8 presents the overall comparison on the test sets. For each treebank, we apply the same sentence segmentation and tokenization used by each best system.\(^11\) We see that our approach outperforms the baseline models on both languages. For Kazakh, our model (with transliteration) achieves a competitive LAS (28.23), which would be the second position in the shared task ranking. As comparison, the best system for Kazakh (Smith et al., 2018) trained a multi-treebank model with four source treebanks, while we only use one source treebank. Their system use predicted POS as input, while ours depends solely on words and characters. The use of more treebanks and predicted POS is beyond the scope of our paper, but it is interesting that our approach can achieve the second best with such minimal resources. For Galician, our best approach outperforms baseline by 8.09 LAS points. Note that, Galician treebank does not come with training data. We use 50:50 train/dev split, while other teams might use higher split for training (for example, the best system (Straka, 2018) uses 90:10 train/dev split). Since we treat Galician as our test data, we did not tune on the proportion for training data, but we guess that this is the main reason why our system achieve rank 10 out of 27.

Compared to cross-lingual models with fastText embeddings (fastText vs. MORPH), we observe that our approach achieve better or comparable performance, showing its potential when there is

---


\(^10\) If a word occurs in both source and target, we use the word embedding of the source language.

\(^11\) UD shared task only provides unsegmented (i.e., sentence-level and token-level) raw test data. However, participants were allowed to use predicted segmentation and tokenization provided by the baseline UDPipe model.
not enough monolingual data available for training word embeddings.

6 Conclusions

In this paper, we investigated various low-resource parsing scenarios. We demonstrate that in the extremely low-resource setting, data augmentation improves parsing performance both in monolingual and cross-lingual settings. We also show that transfer learning is possible with lexicalized parsers. In addition, we show that transfer learning between two languages with different writing systems is possible, and future work should consider transliteration for other language pairs.

While we have not exhausted all the possible techniques (e.g., use of external resources (Rasooli and Collins, 2017; Rosa and Mareček, 2018), predicted POS (Ammar et al., 2016), multiple source treebanks (Lim et al., 2018; Stymne et al., 2018), among others), we show that simple methods which leverage the linguistic annotations in the treebank can improve low-resource parsing. Future work might explore different augmentation methods, such as the use of synthetic source treebanks (Wang and Eisner, 2018) or contextualized language model (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2018) for scoring the augmented data (e.g., using perplexity).

Finally, while the techniques presented in this paper might be applicable to other low-resource languages, we want to also highlight the importance of understanding the characteristics of languages being studied. For example, we showed that although North Sami and Finnish do not share vocabulary, cross-lingual training is still helpful because they share similar syntactic structures. Different language pairs might benefit from other types of similarity (e.g., morphological) and investigating this would be another interesting future work for low-resource dependency parsing.

Acknowledgments

Clara Vania is supported by the Indonesian Endowment Fund for Education (LPDP), the Centre for Doctoral Training in Data Science, funded by the UK EPSRC (grant EP/L016427/1), and the University of Edinburgh. Anders Søgaard is supported by a Google Suggested Research Award. We thank Aibek Makazhanov for helping with Kazakh transliteration, and Miryam de Lhoneux for parser implementation. We also thank Andreas Grivas, Maria Corkery, Ida Szubert, Gozde Gul Sahin, Sameer Bansal, Marco Damonte, Naomi Saphra, Nikolay Bogoychev, and anonymous reviewers for helpful discussion of this work and comments on previous drafts of the paper.

References


A Effects of Fine-Tuning for Cross-Lingual Training

For our cross-lingual experiments in Section 2.3, we observe that fine-tuning on the target treebank always improves parsing performance. Table 9 reports LAS for cross-lingual models with and without fine-tuning.

<table>
<thead>
<tr>
<th>size</th>
<th>cross-base</th>
<th>+Morph</th>
<th>+Nonce</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{100}$</td>
<td>57.9 (+4.6)</td>
<td>59.5 (+6.2)</td>
<td>59.3 (+6.0)</td>
</tr>
<tr>
<td>$T_{50}$</td>
<td>48.3 (+5.8)</td>
<td>49.8 (+7.3)</td>
<td>50.1 (+7.6)</td>
</tr>
<tr>
<td>$T_{10}$</td>
<td>29.8 (+11.3)</td>
<td>34.9 (+16.4)</td>
<td>34.8 (+16.3)</td>
</tr>
</tbody>
</table>

↓ with fine tuning (FT) ↓

<table>
<thead>
<tr>
<th>size</th>
<th>cross-base</th>
<th>+Morph</th>
<th>+Nonce</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{100}$</td>
<td>61.3 (+8.0)</td>
<td>60.9 (+7.6)</td>
<td>61.7 (+8.4)</td>
</tr>
<tr>
<td>$T_{50}$</td>
<td>52.0 (+9.5)</td>
<td>51.7 (+9.2)</td>
<td>52.0 (+9.5)</td>
</tr>
<tr>
<td>$T_{10}$</td>
<td>34.7 (+16.2)</td>
<td>37.3 (+18.8)</td>
<td>35.4 (+16.9)</td>
</tr>
</tbody>
</table>

Table 9: Effects of fine-tuning on North Sámi development data, measured in LAS. mono-base and cross-base are models without data augmentation. % improvements over mono-base shown in parentheses.

B Cyrillic to Latin Alphabet mapping

We use the following character mapping for Cyrillic to Latin Kazakh treebank transliteration.

<table>
<thead>
<tr>
<th>Cyrillic</th>
<th>Latin</th>
<th>Cyrillic</th>
<th>Latin</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>А</td>
<td>Ф</td>
<td>f</td>
</tr>
<tr>
<td>Ø</td>
<td>О</td>
<td>Щ</td>
<td>Sh</td>
</tr>
<tr>
<td>Б</td>
<td>Б</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>В</td>
<td>В</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Г</td>
<td>Г</td>
<td>Щ</td>
<td>Sh</td>
</tr>
<tr>
<td>Д</td>
<td>Д</td>
<td>Щ</td>
<td>Sh</td>
</tr>
<tr>
<td>Е</td>
<td>Е</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Ж</td>
<td>Ж</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>З</td>
<td>З</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>И</td>
<td>И</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Й</td>
<td>Й</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>К</td>
<td>К</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Л</td>
<td>Л</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>М</td>
<td>М</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Н</td>
<td>Н</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>О</td>
<td>О</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>П</td>
<td>П</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>С</td>
<td>С</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Т</td>
<td>Т</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>У</td>
<td>У</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Ф</td>
<td>Ф</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Ц</td>
<td>Ц</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Ш</td>
<td>Ш</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Ш</td>
<td>Ш</td>
<td>Ь</td>
<td>y</td>
</tr>
<tr>
<td>Щ</td>
<td>Щ</td>
<td>Ь</td>
<td>y</td>
</tr>
</tbody>
</table>

Figure 3: Cyrillic to Latin alphabet mapping.