# Improving Code-Switching Dependency Parsing with Semi-Supervised Auxiliary Tasks

**Şaziye Betül Özateş**[1,2], **Arzucan Özgür**[1], **Tunga Gungor**[1], **Özlem Çetinoğlu**[2]

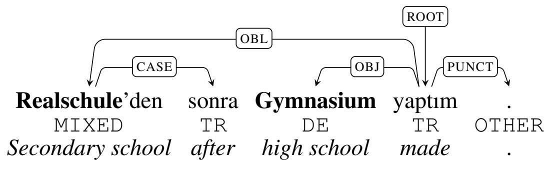
[1] Computer Engineering, Boğaziçi University, Turkey
[2] IMS, University of Stuttgart, Germany
{saziye.bilgin,arzucan.ozgur,gungort}@boun.edu.tr
ozlem@ims.uni-stuttart.de

## Abstract

Code-switching dependency parsing stands as a challenging task due to both the scarcity of necessary resources and the structural difficulties embedded in code-switched languages. In this study, we introduce novel sequence labeling models to be used as auxiliary tasks for dependency parsing of code-switched text in a semi-supervised scheme. We show that using auxiliary tasks enhances the performance of an LSTM-based dependency parsing model and leads to better results compared to an XLM-R-based model with significantly less computational and space complexity. As the first study that focuses on multiple code-switching language pairs for dependency parsing, we acquire state-of-the-art scores on all of the studied languages. Our best models outperform the previous work by 7.4 LAS points on average.

## 1 Introduction

Code-switching (CS) is the producing of utterances by combining phrases and word forms from multiple languages. This is a phenomenon observed frequently in utterances of bilingual speakers (Auer and Wei, 2007). Figure 1 shows an example to this type of utterance formation. Although much work has been done on the syntactic parsing of monolingual languages, CS language pairs are quite understudied in this regard. There have been only a few studies on CS dependency parsing (Bhat et al., 2017; Partanen et al., 2018b; Braggaar and van der Goot, 2021), each focusing only on a single CS language pair. Although CS dependency parsing also benefited from the recent rise of multilingual and cross-lingual natural language processing (NLP) models as shown by van der Goot et al. (2021), these models, which are usually trained on monolingual corpora, are insufficient on CS parsing. The poor performance on CS language pairs is not only due to the lack or scarcity of the training data but also because of the shortage on resources required by deep neural models such as pretrained embeddings, language models, or even raw data. In addition, each language composing a CS language pair inherits its own structural difficulties which contributes a good deal to the problem.


ROOT
OBL
CASE
OBJ
PUNCT
**Realschule**'den sonra **Gymnasium** yaptım .
MIXED TR DE TR OTHER
*Secondary school* *after* *high school* *made* .

'After secondary school I went to high school.'

Figure 1: Dependency tree of a code-switched sentence from the Turkish-German SAGT Treebank. Language ID of each token is located below the token. TR stands for *Turkish*, DE for *German*, MIXED for tokens with intra-word code-switching, OTHER is for punctuation. German tokens and token parts are shown in bold.

Recently, a small number of CS treebanks were manually annotated within Universal Dependencies (UD) (Nivre et al., 2016). Even though these treebanks have little to no training data, their existence provides an opportunity to study dependency parsing also on CS language pairs.

In such low-resource scenarios, utilizing raw data can be helpful in boosting the performance. A common method to benefit from raw data is self-training (McClosky et al., 2006), a semi-supervised approach where a small number of labeled data is used to train a model that is later used to predict labels for unlabeled data. This pseudo-labeled data is then combined with the initial data to re-train the model. This method is usually found successful in low-resource scenarios (Rybak and Wróblewska, 2018; Yu et al., 2020), although error propagation is a known problem when pseudo-labels are noisy.

With very restricted resources, we hypothesize that CS dependency parsing can also benefit from unlabeled data. Based on this hypothesis, we form our *first research question*: is using pseudo-labeled data directly beneficial for CS dependency parsing or can we find better ways of integrating the knowledge from pseudo-labeled data?

Starting from this question, we follow a deep contextualized self-training approach (Rotman and Reichart, 2019) and integrate semi-supervised auxiliary tasks to the parsing architecture to enhance CS dependency parsing. Our method enhances a widely-used BiLSTM-based parser (Dozat and Manning, 2017) by training parsing-related auxiliary sequence labeling tasks on automatically labeled data and combining these trained auxiliary task models with the base parser through a gating mechanism. We introduce new sequence labeling tasks that are shown to be beneficial in improving the parsing performance. Seeing the success of our semi-supervised enhancement method on the BiLSTM-based parser, we form our *second research question*: can we reach even better parsing scores if we combine this enhancement method with XLM-R (Conneau et al., 2020), a state-of-the-art (SOTA) transformer-based language model that shows superior performance on many NLP tasks? Our experimental results demonstrate notable success of our proposed models over the previous state-of-the-art on these treebanks. Our contributions are as follows:

- We employ a semi-supervised learning approach based on auxiliary tasks for CS dependency parsing. We present the first study with a focus on parsing all CS UD treebanks and achieve SOTA results on all of them.
- We introduce novel sequence labeling tasks including a CS-specific one, that capture syntactic information better and hence improve dependency parsing.
- We adapt this method to the powerful XLM-R model and elaborate the effectiveness of this approach when combined with XLM-R-based word representation for dependency parsing. We demonstrate that the mighty transformer model remains inadequate for the case of low-resource CS parsing.

## 2 Related Work

Code-switching dependency parsing is a newly-studied research area. The first CS UD treebank was created by Bhat et al. (2017) which included only a test set of Hindi-English sentences. In the absence of CS training data, the test set was split to monolingual fragments and existing Hindi and English monolingual treebanks in UD were used to parse these fragments. Bhat et al. (2018) extended this dataset with a CS training set. They trained a BiLSTM architecture on this additional training data by also integrating syntactic knowledge extracted from monolingual treebanks.

Partanen et al. (2018b) laid the first foundations of a Komi-Russian UD treebank with 25 CS sentences. They adopted a multilingual parsing approach (Lim and Poibeau, 2017) and used Russian and Komi monolingual training data with bilingual Komi-Russian word embeddings. Later, this treebank expanded into the Komi-Zyrian IKDP treebank (Partanen et al., 2018a).

Çetinoğlu and Çöltekin (2019) created a Turkish-German UD treebank from a Turkish-German spoken corpus. Seddah et al. (2020) introduced the Maghrebi Arabic-French treebank and performed parsing experiments on the treebank using UDPipe (Straka and Straková, 2017). This treebank is yet to be included in the UD. A Frisian-Dutch UD treebank which includes only test data was introduced by Braggaar and van der Goot (2021). The authors performed data selection from eight related monolingual treebanks using Latent Dirichlet Allocation (Blei et al., 2003) to create a training set. Their experiments performed using a deep biaffine parser (van der Goot et al., 2021) demonstrated no significant performance difference between training the parser on the selected training set and only on a Dutch monolingual treebank.

Lately, multilingual and cross-lingual parsing studies have begun to include CS treebanks in their experimental setups. van der Goot et al. (2021) presented a multi-task learning tool that utilizes multilingual BERT (Devlin et al., 2019) to perform several NLP tasks, including dependency parsing. Evaluation was done on all available UD treebanks which include CS UD treebanks mentioned above. The model was fine-tuned on training set of each treebank, which is also the case for Hindi-English and Turkish-German CS treebanks. For Frisian-Dutch and Komi-Russian CS treebanks with no training data, they used Dutch Alpino and Russian SynTagRus treebanks, respectively. Müller-Eberstein et al. (2021) applied a sentence level genre-based data selection from UD treebanks in a cross-lingual setup. They trained a multilingual BERT-based biaffine parser (van der Goot et al., 2021) for 12 low-resource UD treebanks including Hindi-English and Turkish-German CS treebanks.

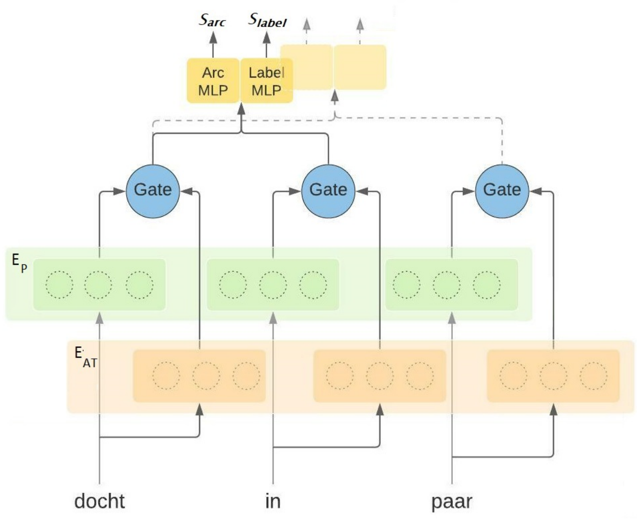


Figure 2: The parser architecture with semi-supervised auxiliary task enhancement. $E_p$ is the parser encoder, $E_{AT}$ is the sequence labeler encoder trained on one of the auxiliary tasks. For a given token pair, the model calculates a weighted average of each token's hidden representation from $E_P$ and $E_{AT}$. The resulting vectors are given to two multi-layer perceptrons (MLP) to produce an arc score $S_{arc}$ and a label score $S_{label}$ for the given token pair. The input tokens are taken from the Frisian-Dutch Fame Treebank.

Our study on CS dependency parsing differs from the previous work in the sense that none of the previous work utilized raw CS data to improve parsing in a semi-supervised scheme.

## 3 Methodology

### 3.1 Base Parsing Model

Our base parser is a neural graph-based parser by Dozat and Manning (2017) that uses two biaffine classifiers, one to predict the head of a given token and the other to predict the resulting arc's label. For input representation, the model uses BiLSTM modules to compute learned word embeddings and add them to their corresponding pretrained word embeddings that are later concatenated with corresponding part-of-speech (POS) embeddings. To ensure a well-formed tree at test time, the maximum spanning tree (MST) algorithm is used.

### 3.2 Semi-supervised Enhancement through Auxiliary Sequence Labeling Tasks

We follow Rotman and Reichart (2019) to exploit unlabeled data for CS dependency parsing. Rather than directly using pseudo-labeled data as an additional source in training, the main idea is to extract and utilize parsing-related knowledge from automatically parsed data. This is achieved by training contextualized embedding models on a number of auxiliary sequence labeling tasks derived from the raw data parsed by the base parser and then combining encoders of these trained models with that of the base parser through a gating procedure (Sato et al., 2017) as described in Section 3.3. Figure 2 depicts this enhanced parser. The combined model is then re-trained on the gold labeled data.

For their experimental setup, Rotman and Reichart (2019) consider three token-level sequence labeling schemes to extract the structural information encoded in the parsed sentences. These are:

**(i) Number of Children (NOC)** The task is to predict the number of children each token has in a dependency tree.

**(ii) Distance to the Root (DTR)** Each token is tagged with its minimum distance to the root token of the dependency tree.

**(iii) Relative POS-based Encoding (RPE)** Each token in a sentence is tagged with its head's POS tag in a simplified form and its distance from the head. The distance calculation considers only the intermediate tokens that share the same POS tag with its head.

Although these three auxiliary tasks offer a comprehensive scheme in terms of extracting parsing-related knowledge from automatically parsed data, we search ways of channeling the embedded knowledge in parsed trees more thoroughly to the trained word embedding layers of the parser. We come up with three additional sequence labeling tasks:

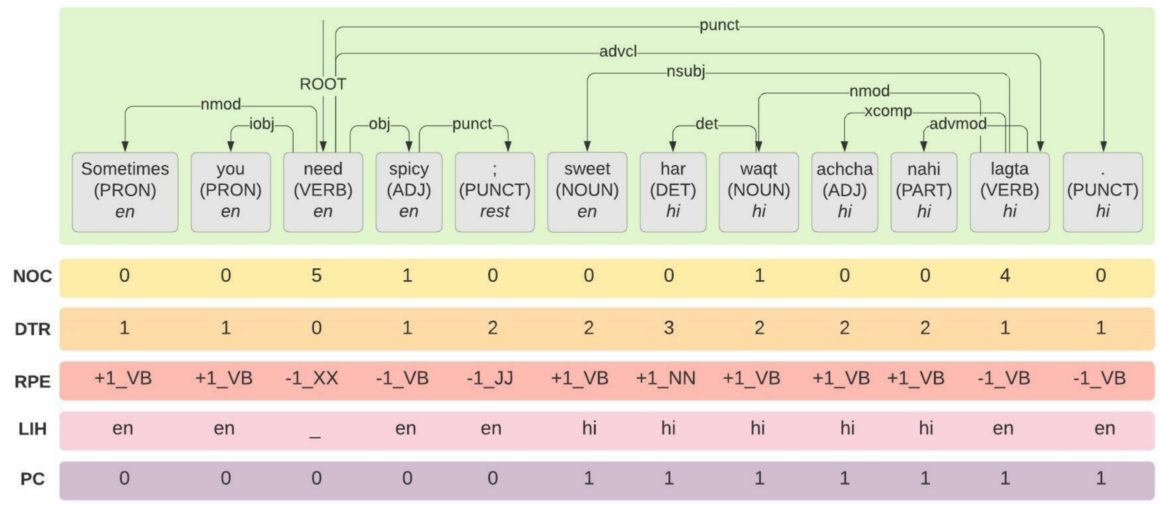


Figure 3: The dependency tree of an example sentence from Hindi-English HIENCS Treebank. Each node in the tree is tagged with the five auxiliary task schemes depicted in Section 3.2. Tags for the case of the SMH scheme are not shown for this example since the HIENCS Treebank does not include morphology.

**(iv) Language ID of Head (LIH)** We start with CS-specific features of parsed trees. The most prominent of them is the language ID (LID) features of the tokens in CS treebanks. Considering the positive impact of LIDs in various other NLP tasks (Jamatia et al., 2015; Aguilar and Solorio, 2020; Özateş and Çetinoğlu, 2021), we design a simple auxiliary sequence labeling task that makes use of LIDs. Unlike previous work using token LIDs, LIH tags each token with the LID of its head. This way, information about the language of tokens with which each token tends to relate in terms of dependencies is conveyed to the learning model.

**(v) Simplified Morphology of Head (SMH)** Morphological features are found to be beneficial in parsing morphologically-rich languages (Dehouck and Denis, 2018). This was our motivation to create a new auxiliary task based on morphology. In the SMH scheme, each token is assigned its head's morphological features. To reduce the number of labels, we use only a subset of the morphological features set, selected by considering the inclusiveness and the prevalence of the features across the data.[1] The main idea of SMH is to provide morphological clues to the parser while also giving information about the structure of the tree.

A similar approach is also tried by Sandhan et al. (2021). They define a sequential task to predict the full set of morphological features for a given token. In our preliminary experiments, we observed that using the full set of morphological features does not improve the accuracy. In CS treebanks the unique number of features is increased due to the combination of language-specific feature sets of the language pair, making the task more complex. To reduce the complexity, we design SMH as utilizing only a subset of the morphological features of (not the token itself, but) the head of the token.

**(vi) Punctuation Count (PC)** Lastly, we design the PC task that only needs root tokens unlike all other tasks that need parsed trees to function. PC is also not dependent on morphological, POS, or LID tags as SMH, RPE, and LIH tasks.

PC simply tags each token with the number of punctuations between that token and the root token in the sentence. We observe a connection between the position of punctuation and phrase boundaries in a sentence which goes in line with previous studies (Li et al., 2010; Spitkovsky et al., 2011). PC roughly groups tokens into phrases that usually constitute sub-trees in a dependency tree.

Figure 3 shows the outputs of these tasks on the dependency tree of an example CS sentence.

## 3.3 The Gating Procedure

To create the final parser, the trained auxiliary task models are combined with the base parser through a gating mechanism (Sato et al., 2017) which learns to scale between the encoders of the auxiliary sequence labelers and that of the parser (see Fig. 2).

[1]The selected UD features are *Aspect, Case, Foreign, Mood, NumType, Person,* and *VerbForm.*

Formally, the combined representation can be formulated as:

$$b^t = \sigma(W^{gate}(e^{parser} \oplus e^{labeler}) + w^{gate})$$
$$g^t = b^t \cdot e^{parser} + (1 - b^t) \cdot e^{labeler}$$

where $e^{parser}$ and $e^{labeler}$ are the outputs of the parser and sequence labeler encoders, respectively. $\oplus$ denotes concatenation. $W^{gate}$ and $w^{gate}$ are the learned parameters of the gating procedure and $\sigma$ is the sigmoid function. The final combined vector $g^t$ is then given to the biaffine classifiers.

### 3.4 Transformer-based Adaptation of the Model

Our base parser as described in Dozat and Manning (2017) has some shortcomings in the choice of the input representation, especially when the target language has very little or no training data and there is no accompanying pretrained word embeddings to represent the input. This is also the case with CS language pairs. In that situation, utilizing the expressive power of transformers can be a good solution. Pretrained on huge amounts of raw data in different languages, multilingual transformer-based language models have proven remarkably effective (Devlin et al., 2019; Sanh et al., 2019; Liu et al., 2019). One such model is XLM-R (Conneau et al., 2020). Pretrained on text data in 100 languages, XLM-R shows SOTA performance in many languages including low-resource ones.

To the best of our knowledge, such a deep contextualized semi-supervised scheme has not been incorporated with XLM-R before. So, we re-implement the auxiliary task modules and the combined parsing approach for an XLM-R-based encoding module. For this purpose we follow the XLM-R-based parsing architecture of Grünewald et al. (2021) which has the same biaffine parsing model described in Dozat and Manning (2017). Our aim is to observe how extracting parsing-related knowledge from semi-supervised auxiliary tasks affects a multilingual transformer model.

## 4 Experiments

### 4.1 Data

We perform experiments on all CS treebanks[2] in Universal Dependencies (v2.8).[3] These are Komi-Zyrian IKDP (Kpv-Ru), Hindi-English HIENCS (Hi-En), Frisian-Dutch Fame (Fy-Nl), and Turkish-German SAGT (Tr-De) treebanks. All except Hi-En are based on spoken CS data. Hi-En is constructed from bilingual tweets. Table 1 states basic statistics and related resources for each treebank.

### 4.2 Training Setup

Due to lack of training data in some CS treebanks, we have two types of experimental setup. We train the parser models on *in-domain* data for Hi-En and Tr-De. In these experiments we use each treebank's own training set. However, Kpv-Ru and Fy-Nl consist of a test set only. Hence, training of the latter two treebanks are on *out-of-domain* data. For Kpv-Ru which includes Komi-Russian code-switching, we train the models on Komi-Zyrian Lattice UD Treebank (Partanen et al., 2018a) of monolingual Komi data. The first 562 sentences in Komi-Zyrian Lattice are used for training, the remaining 100 are used for development. For Fy-Nl, our training data is the Dutch Alpino UD Treebank (Van der Beek et al., 2002). We chose Dutch Alpino over the other Dutch UD treebank (LassySmall) as Alpino is found more effective in parsing Fy-Nl (Braggaar and van der Goot, 2021).

#### 4.2.1 Unlabeled Data

**Komi-Russian** Komi Social Media Corpus[4] is part of a social media corpora project for minority Uralic languages (Arkhangelskiy, 2019). The data is crawled from *vkontakte*, a social media service mostly popular in Russia. Collected texts are automatically separated to monolingual segments of *Komi, Russian*, or *Unknown* via a dictionary-based method. For our purposes, we extract 3,862 CS sentences from the corpus by joining consecutive segments that alternate between *Komi* and *Russian*.

**Hindi-English** We employ the datasets in the LinCE CS benchmark[5] (Aguilar et al., 2020) for this language pair. The benchmark provides three different corpora with gold LID and POS labels for Hindi-English (Mave et al., 2018; Singh et al., 2018a,b). We combine these three corpora to use them as unlabeled data. The resulting data consists of 10,989 sentences.

[2]There is also the Maghrebi Arabic-French Treebank (Seddah et al., 2020) but its annotations are not yet compatible with the UD scheme and it is not included in the UD repository.

[3]https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-3687. All UD treebanks used in this paper are licensed with CC BY-SA 4.0.

[4]Available for research purposes. We obtained the corpus by contacting Timofey Arkhangelskiy at http://komi-zyrian.web-corpora.net/index_en.html

[5]The datasets are publicly available for research purposes.

|  | Kpv-Ru (Komi-Russian) | Hi-En (Hindi-English) | Fy-Nl (Frisian-Dutch) | Tr-De (Turkish-German) |
|---|---|---|---|---|
| Train | – | 1,448 | – | 578 |
| Dev | – | 225 | 150 | 801 |
| Test | 214 | 225 | 250 | 805 |
| CMI | 16.97 | 36.08 | 17.80 | 28.78 |
| Morphology | yes | no | no | yes |
| Monolingual treebanks | both | both | only Dutch | both |
| Unlabeled CS data | Komi Social Media | LinCE | FAME! | TuGeBiC |
| XLM-R | only Russian | both | both | both |
| FastText | only Russian | both | both | both |

Table 1: Some statistics and related resources for the CS treebanks. Fy-Nl is provided as a single test set of 400 utterances. As in Braggaar and van der Goot (2021), we split it into a development set (first 150 utterances) and a test set (remaining 250 utterances). CMI is the code-mixing index (Das and Gambäck, 2014) that shows how frequent code-switching happens in the text.

**Frisian-Dutch** We extract CS sentences from the FAME! Corpus[6] (Yılmaz et al., 2016) which contains radio broadcasts in Frisian-Dutch. From this corpus, which is also the source of the Fy-Nl treebank, we select 2,170 sentences that include at least one CS point and are not already in the treebank.

**Turkish-German** TuGeBiC[7] (Treffers-Daller and Çetinoğlu, 2022) is a set of transciptions, collected from interviews with Turkish-German bilinguals in the 90s (Treffers-Daller, 2020). It contains 16,950 sentences. We use the whole corpus, and only remove the speaker IDs and metadata from the files.

#### 4.2.2 Sequence Labeler Training

Training auxiliary models on sequence labeling tasks is done on automatically parsed version of the corresponding unlabeled data for each treebank. Some of the sequence labeling tasks need specific labels on unlabeled data to function. These are POS tags for RPE, LID labels for LIH, and morphological annotation for SMH. In training of these tasks, we use gold labels when available (POS tags for Hi-En; LIDs for Kpv-Ru, Hi-En, and Fy-Nl) and train taggers in the absence of gold labels (POS tags for Kpv-Ru, Fy-Nl, and Tr-De; LIDs for Tr-De; morphological features for Kpv-Ru and Tr-De).

### 4.3 Baselines

As our baseline, we use Ma et al. (2018)'s re-implementation of the biaffine parser by Dozat and Manning (2017). We call this model $\texttt{Base}_{\texttt{LSTM}}$ as it uses BiLSTMs for contextualized word vectors.

As a second baseline, we implement the traditional self-training approach (McClosky et al., 2006) in which the parser is first trained only on gold labeled data. Then, labels of unlabeled data are predicted by the trained parser. Finally the parser is re-trained on the combination of gold labeled data and pseudo-labeled data. We name this approach as `Self-training`.

For our experiments with XLM-R, we use Grünewald et al. (2021)'s implementation of the biaffine parser with XLM-R-based input representation. Input word embeddings are calculated as a weighted sum of all intermediate outputs of the transformer layers. Coefficients of the weighted sum are learned during the training phase. Apart from its multilingual transformer-based contextualized word representation model, it has the same biaffine parsing model in Dozat and Manning (2017). We call this version $\texttt{Base}_{\texttt{XLMR}}$.

Hyper-parameters of both parser models and sequence labelers can be found in Appendix A.1.

### 4.4 Semi-supervised Enhancement Models

We provide the list of enhancement models built on top of $\texttt{Base}_{\texttt{LSTM}}$ and $\texttt{Base}_{\texttt{XLMR}}$ where parser is combined with a sequence labeler trained on:

- `+NOC`: *Number of Children*,
- `+DTR`: *Distance to the Root*,
- `+RPE`: *Relative POS Encoding*,
- `+LIH`: *Language ID of Head*,
- `+SMH`: *Simplified Morphology of Head*,[8]
- `+PC`: *Punctuation Count*.[9]

[6]Available via a license agreement. `https://www.ru.nl/clst/tools-demos/datasets/`

[7]Available at `https://github.com/ozlemcek/TuGeBiC`

[8]Note that only Kpv-Ru and Tr-De treebanks have morphological annotation. Hence, `+SMH` is applied only to them.

[9]The `+PC` model is not applied to Fy-Nl since the treebank does not have punctuation.

Additionally, we perform experiments by ensembling more than one auxiliary task model with the base parser. We experiment with two configurations. First, we integrate *Number of Children*, *Distance to the Root*, and *Relative POS Encoding* models together (+NOC, +DTR, +RPE). This is also the ensemble configuration in Rotman and Reichart (2019). Since we have additional three tasks, we also make the combination of three best performing models for each treebank and name this ensemble version as +Best Combination.[10] For combining encoders of more than one auxiliary task model with the parser encoder, we use Rotman and Reichart (2019)'s extension to the gating mechanism of Sato et al. (2017).

We perform three runs for each model and report the average scores. We measure the performance of all models using the CoNLL 2018 Shared Task evaluation script[11] and report the unlabeled and labeled attachment scores (UAS and LAS, respectively).

## 5 Results and Discussion

Table 2 shows the performance of all LSTM-based models and of the previous works on the test set of each treebank in terms of attachment scores. Significance testing is performed using the approximate randomization test (Noreen, 1989) on the model outputs with the number of shuffles set to 5,000.

**Comparison to Baselines** On all treebanks, the auxiliary task enhancement methods improve the scores when compared to $\text{Base}_{\text{LSTM}}$ by 4.94 points in UAS and 3.86 points in LAS on average. The best performing enhancement model differs across treebanks. We observe the same pattern for the traditional self-training method. Self-training fails to surpass the proposed approach on any of the treebanks. Its parsing performance even falls below that of $\text{Base}_{\text{LSTM}}$ on Kpv-Ru and Tr-De. It shows the highest improvement with respect to $\text{Base}_{\text{LSTM}}$ on Fy-Nl. Yet, the best one of the auxiliary task enhancement methods significantly outperforms Self-training on each treebank.

**New Individual Tasks** The +LIH model which employs LIDs performs best on Kpv-Ru, and second best on Hi-En. Its performance on Tr-De and Fy-Nl is comparable with the other models. It is also in the Best Combination ensemble for all treebanks. This indicates the importance of language IDs in CS dependency parsing.

The +SMH model which is only applied to Kpv-Ru and Tr-De is the best performing one on Tr-De. However, all other tasks outperform +SMH on Kpv-Ru. This might be due to the quality difference in morphological taggers trained on these treebanks. The morphological tagger we trained on the CS training set of Tr-De has an accuracy of 82% on its test set. However, to train a tagger for Kpv-Ru we used monolingual Komi data only. Accuracy of this tagger on Kpv-Ru test set is 66%. It seems the Kpv-Ru parser suffers from error propagation.

The simplest enhancement model +PC performs comparable to others, even outperforming +NOC and +DTR on Kpv-Ru and Tr-De. Since it only needs the root position in the sentence to perform, this model can be an alternative to other models when gold/predicted POS or morphological tags are hard to acquire. It can also be preferred when the error propagated to the auxiliary tasks from the base parser through predicted trees is high, damaging accuracy of the tasks that rely on these parses.

**Individual Tasks vs Ensembles** Ensembling multiple tasks improves UAS and LAS on Hi-En and Fy-Nl and LAS on Tr-De when compared with the best performing single task. The +Best Combination ensemble works better on Fy-Nl and Tr-De than the +NOC, +DTR, +RPE ensemble proposed by Rotman and Reichart (2019). Looking at the overall results, we observe that including +RPE and +LIH together has a favorable effect on improving CS parsing performance.

**Who Benefits Most and Least?** Fy-Nl is the most benefited treebank from the proposed model. The best performing enhancement model +Best Combination on Fy-Nl achieves almost 10/7 points increase in UAS/LAS when compared with $\text{Base}_{\text{LSTM}}$. The least benefited treebank is Kpv-Ru with 2.5/1.1 points increase in UAS/LAS. Having similar amount of unlabeled data and no CS training data, these treebanks differ in their training data amounts. The Dutch Alpino Treebank used to train Fy-Nl models has 13,603 sentences whereas the Komi-Zyrian Lattice Treebank for Kpv-Ru models includes 662 sentences. So, automatic parsing of unlabeled data of Kpv-Ru by a model trained on 662 sentences can be much noisier than that of Fy-Nl. In Appendix A.2, we show that the performance ranking of the systems does not change by the amount of gold training data.

[10] Due to high memory consumption of XLM-R-based models, this ensemble technique cannot be applied to our XLM-R-based parsing architecture.

[11] https://universaldependencies.org/conll18/conll18_ud_eval.py

| | | Kpv-Ru | | Hi-En | | Fy-Nl | | Tr-De | |
|---|---|---|---|---|---|---|---|---|---|
| | | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS |
| *Baselines* | Base$_{\text{LSTM}}$ | 62.24 | 45.10 | 80.10 | 71.29 | 64.97 | 49.56 | 67.50 | 57.88 |
| | Self-training | 59.55 | 43.27 | 80.47 | 72.88 | 68.91 | 53.24 | 60.86 | 52.04 |
| *Semi-supervised Enhancement* | +NOC | 64.83* | 46.53* | 81.67 | 72.94 | 71.80* | 53.35 | 70.86* | 60.97* |
| | +DTR | 64.80* | 45.53 | 81.94 | 72.96 | 71.48* | 53.10 | 70.88* | 60.63* |
| | +RPE | 64.95* | 45.90 | 82.75* | 73.84 | 72.98* | 54.12 | 71.40* | 61.46* |
| | +LIH | **65.70*** | **47.13*** | 82.24* | 73.54 | 72.20* | 51.98 | 71.39* | 61.46* |
| | +SMH | 64.63* | 45.31 | - | - | - | - | **71.41*** | 61.50* |
| | +PC | 64.67* | 46.79* | 81.40 | 72.76 | - | - | 71.25* | 61.44* |
| *Ensemble* | +NOC,+DTR,+RPE | 65.59* | 46.86* | 82.75* | **74.09*** | 73.97* | 56.10* | 70.55* | 60.95* |
| | +Best Combination† | 64.98* | 46.22* | **82.77*** | 74.02* | **74.69*** | **56.39*** | 70.92* | **61.65*** |
| *Previous Work* | Bhat et al. (2018) | - | - | 80.23 | 71.03 | - | - | - | - |
| | Braggaar and van der Goot (2021) | - | - | - | - | 70.20 | 55.60 | - | - |
| | van der Goot et al. (2021) | - | 22.20 | - | 65.50 | - | 54.00 | - | 60.90 |
| | Müller-Eberstein et al. (2021) | - | - | 73.62 | 62.66 | - | - | 66.75 | 55.04 |

Table 2: Attachment scores of baselines, our models, and the previous works on all CS UD treebanks. +SMH is not applicable to Hi-En and Fy-Nl due to the lack of morphology in these treebanks. +PC cannot be applied to Fy-Nl since it has no punctuation. †Best combination for each treebank: +NOC,+LIH,+PC for Kpv-Ru, +DTR,+RPE,+LIH for Hi-En, +NOC,+RPE,+LIH for Fy-Nl, and +RPE,+LIH,+SMH for Tr-De. The best scores for each dataset are underlined and bold. Scores marked with $*$ significantly outperform both Base$_{\text{LSTM}}$ and Self-training.

| | Kpv-Ru | | Hi-En | | Fy-Nl | | Tr-De | |
|---|---|---|---|---|---|---|---|---|
| | UAS | LAS | UAS | LAS | UAS | LAS | UAS | LAS |
| Base$_{\text{XLMR}}$ | 57.90 | 43.12 | 81.42 | 71.54 | 65.75 | 50.27 | **75.93** | **66.30** |
| +NOC | 57.09 | 42.79 | 81.28 | 71.58 | **67.50*** | **51.64*** | 75.79 | 65.98 |
| +DTR | 56.65 | 42.37 | **82.15*** | 71.89 | 66.85* | 50.45 | 75.56 | 65.73 |
| +RPE | **58.77*** | **43.84** | 81.79 | 71.84 | 67.35* | 51.13* | 75.49 | 65.77 |
| +LIH | 57.24 | 43.19 | 81.92 | **71.93** | 66.26 | 50.10 | 75.51 | 65.78 |
| +SMH | 56.98 | 43.25 | - | - | - | - | 75.53 | 65.66 |
| +PC | 56.81 | 41.97 | 81.46 | 71.89 | - | - | 75.14 | 65.45 |

Table 3: Performance of XLM-R-based parser and our XLM-R adaptation of auxiliary task enhancement models. The best scores for each dataset are underlined and bold. Scores marked with $*$ significantly outperform Base$_{\text{XLMR}}$.

**Comparison to Previous Work** The best enhancement model always achieves better scores than previous state-of-the-art on each treebank. In this respect, the biggest improvement is observed on Kpv-Ru with more than 24 points increase in LAS. In addition, it should be noted that model architectures are not quite comparable as some of the previous work use a lot more resources than our models. For instance, Müller-Eberstein et al. (2021) perform data selection on whole UD datasets for training and utilize multilingual BERT.

**Proposed Method and XLM-R** Attachment scores of Base$_{\text{XLMR}}$ and our XLM-R adaptation of auxiliary task enhancement models are given in Table 3. Our first observation is the limited performance of Base$_{\text{XLMR}}$ in parsing CS treebanks. We see that the enhancement models do not have the same impact on Base$_{\text{XLMR}}$ as they have on Base$_{\text{LSTM}}$. The only significant performance increase is on Fy-Nl where the best performing enhancement model +NOC outperforms Base$_{\text{XLMR}}$ by almost 2/1.5 points in UAS/LAS. For Kpv-Ru, the only model that surpasses the baseline is +RPE. The difference is found statistically significant only in UAS. For Hi-En, all enhancement models except +NOC perform better than Base$_{\text{XLMR}}$. Yet, the only significant improvement is achieved by +DTR in UAS. None of the enhancement models surpass Base$_{\text{XLMR}}$ on Tr-De but the difference between the scores is not found to be significant. Another remarkable observation is our models built on top of Base$_{\text{LSTM}}$ outperforming all XLM-R-based models with the exception of Tr-De. This answers our second research question: XLM-R is not always the best option. For powerful models like XLM-R, multilinguality can harm the performance when the target language is unknown to the model. Our results suggest that in such cases it is better to employ simpler models that are tailored for the exact task.

**Comparison of the Proposed Approach with Baselines in terms of Computational Resources** Table 4 provides time and memory usage of $\texttt{Base}_{\texttt{LSTM}}$, $\texttt{Base}_{\texttt{XLMR}}$, and our proposed best model for each treebank. Labeled attachment scores (LAS) acquired by these models on each treebank are also given.

| | Kpv-Ru | Hi-En | Fy-Nl | Tr-De |
|---|---|---|---|---|
| *Training time* | | | | |
| $\texttt{Base}_{\texttt{LSTM}}$ | **0h9m** | **0h15m** | **0h45m** | **0h20m** |
| Our best model | 0h25m | 0h40m | 2h30m | 0h55m |
| $\texttt{Base}_{\texttt{XLMR}}$ | 3h40m | 3h15m | 11h0m | 1h30m |
| *Memory usage (GB)* | | | | |
| $\texttt{Base}_{\texttt{LSTM}}$ | **3.6** | **3.6** | **3.8** | **3.5** |
| Our best model | 4.5 | 7.6 | 7.3 | 7.4 |
| $\texttt{Base}_{\texttt{XLMR}}$ | 9.9 | 7.9 | 9.6 | 8.4 |
| *LAS* | | | | |
| $\texttt{Base}_{\texttt{LSTM}}$ | 45.10 | 71.29 | 49.56 | 57.88 |
| Our best model | **47.13** | **74.09** | **56.39** | 61.65 |
| $\texttt{Base}_{\texttt{XLMR}}$ | 43.12 | 71.54 | 50.27 | **66.30** |

Table 4: Comparison of baselines and the proposed approach according to training time, memory usage during training, and LAS. Our best model on Kpv-Ru is the `+LIH` model. For all other treebanks, our best model is an ensemble that combines three task models.

From the table, we observe that there is a trade-off between performance and resource consumption for the three models. The training time of the $\texttt{Base}_{\texttt{LSTM}}$ model is the shortest. Yet, our best model improves the performance significantly at the expense of a slight increase in training time. $\texttt{Base}_{\texttt{XLMR}}$ has the longest training time by a large margin.

In terms of memory usage, there is a similar pattern to that of training time. $\texttt{Base}_{\texttt{LSTM}}$ needs approximately 50% less memory than our best model, yet there is on average 3.86 points gap between LAS of the two models. $\texttt{Base}_{\texttt{XLMR}}$ is again the least preferable model here due to its highest memory consumption and low performance on parsing the treebanks with the exception of Tr-De. Only for Tr-De it outperforms the other two models and can be the model of choice for the parsing of Tr-De data.

Considering the long training time and high resource consumption of the XLM-R-based parser and the success of our LSTM-based enhancement models, we suggest LSTM-based auxiliary task enhancement for low-resource dependency parsing of CS data.

## 6 Conclusion

In this paper, we focus on CS dependency parsing. We present a semi-supervised auxiliary task enhancement to a graph-based neural parser and create novel sequence labeling tasks that are shown as useful in improving the parser's success. Experimental results show that our enhancement technique achieves SOTA performance on all CS UD treebanks and helps better utilization of unlabeled data for CS dependency parsing. We combine our enhancement models with XLM-R to see their performance on a multilingual transformer-based model. Results demonstrate that the powerful XLM-R shows limited performance and fails to surpass our semi-supervised auxiliary task enhancement models. Our implementation of the proposed sequence labeling tasks and the XLM-R-based enhancement are publicly available for research purposes at `https://github.com/sb-b/ss-cs-depparser`.

## Acknowledgements

This work is funded by DFG via project CE 326/1-1 "Computational Structural Analysis of German-Turkish Code-Switching (SAGT)". The numerical calculations reported in this paper were performed at the computing servers of IMS at University of Stuttgart and at TUBITAK ULAKBIM (TRUBA resources). GEBIP Award of the Turkish Academy of Sciences (to A.Ö.) is gratefully acknowledged. We thank the anonymous reviewers for their valuable comments and feedbacks.

## Ethical Considerations

The UD datasets we use in this study are all from published works and licensed with CC BY-SA 4.0. All the used unlabeled corpora also have accompanying papers and were made publicly available by their providers. To utilize TuGeBiC in our experiments, we removed speaker IDs that were already anonymized.

Code-switching is very understudied computationally and by employing datasets from multiple language pairs, including minority languages, we minimize the potential risk of exposure. Our study asks foundational research questions on code-switching datasets. Our experiments show that even within our datasets results vary, hence we believe the paper avoids possible overgeneralizations.

XLM-R-based models employed in this study have higher memory consumption than the LSTM-based models (see Table 4). They also require longer training time which means higher energy consumption. To reduce this negative side-effect, we avoid performing the same experiment many times and repeated each experiment only three times. Moreover, we chose not to utilize XLM-R-large model with 550M parameters but instead used XLM-R-base model with 270M parameters to reduce the training time and memory consumption. All the experiments with XLM-R models in this study were run on a single NVIDIA RTX A6000 GPU.

## References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Gustavo Aguilar and Thamar Solorio. 2020. From English to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8033–8044, Online. Association for Computational Linguistics.

Timofey Arkhangelskiy. 2019. Corpora of social media in minority Uralic languages. In *Proceedings of the Fifth International Workshop on Computational Linguistics for Uralic Languages*, pages 125–140, Tartu, Estonia. Association for Computational Linguistics.

Peter Auer and Li Wei. 2007. *Handbook of multilingualism and multilingual communication*, volume 5. Walter de Gruyter.

Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. 2017. Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 324–330, Valencia, Spain. Association for Computational Linguistics.

Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. 2018. Universal Dependency parsing for Hindi-English code-switching. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 987–998, New Orleans, Louisiana. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Anouck Braggaar and Rob van der Goot. 2021. Challenges in annotating and parsing spoken, code-switched, Frisian-Dutch data. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 50–58, Kyiv, Ukraine. Association for Computational Linguistics.

Özlem Çetinoğlu and Çağrı Çöltekin. 2019. Challenges of annotating a code-switching treebank. In *Proceedings of the 18th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2019)*, pages 82–90, Paris, France. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed Indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387, Goa, India. NLP Association of India.

Mathieu Dehouck and Pascal Denis. 2018. A framework for understanding the role of morphology in Universal Dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2864–2870, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Stefan Grünewald, Annemarie Friedrich, and Jonas Kuhn. 2021. Applying Occam's razor to transformer-based dependency parsing: What works, what doesn't, and what is really necessary. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 131–144, Online. Association for Computational Linguistics.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed English-Hindi Twitter and Facebook chat messages. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 239–248, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhenghua Li, Wanxiang Che, and Ting Liu. 2010. Improving dependency parsing using punctuation. In *2010 International Conference on Asian Language Processing*, pages 53–56.

KyungTae Lim and Thierry Poibeau. 2017. A system for multilingual dependency parsing based on bidirectional LSTM feature representations. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 63–70, Vancouver, Canada. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Deepthi Mave, Suraj Maharjan, and Thamar Solorio. 2018. Language identification and analysis of code-switched social media text. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 51–61, Melbourne, Australia. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Max Müller-Eberstein, Rob van der Goot, and Barbara Plank. 2021. Genre as weak supervision for cross-lingual dependency parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4786–4802, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. 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 Tenth International Conference on Language Resources and Evaluation (LREC'16), pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Şaziye Betül Özateş and Özlem Çetinoğlu. 2021. A language-aware approach to code-switched morphological tagging. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 72–83, Online. Association for Computational Linguistics.

Niko Partanen, Rogier Blokland, KyungTae Lim, Thierry Poibeau, and Michael Rießler. 2018a. The first Komi-Zyrian Universal Dependencies treebanks. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 126–132, Brussels, Belgium. Association for Computational Linguistics.

Niko Partanen, Kyungtae Lim, Michael Rießler, and Thierry Poibeau. 2018b. Dependency parsing of code-switching data with cross-lingual feature representations. In *Proceedings of the Fourth International Workshop on Computational Linguistics of Uralic Languages*, pages 1–17, Helsinki, Finland. Association for Computational Linguistics.

Guy Rotman and Roi Reichart. 2019. Deep contextualized self-training for low resource dependency parsing. *Transactions of the Association for Computational Linguistics*, 7(0):695–713.

Piotr Rybak and Alina Wróblewska. 2018. Semi-supervised neural system for tagging, parsing and lematization. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 45–54, Brussels, Belgium. Association for Computational Linguistics.

Jivnesh Sandhan, Amrith Krishna, Ashim Gupta, Laxmidhar Behera, and Pawan Goyal. 2021. A little pretraining goes a long way: A case study on dependency parsing task for low-resource morphologically rich languages. In *Proceedings of the 16th*

*Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 111–120, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain Universal Dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79, Vancouver, Canada. Association for Computational Linguistics.

Djamé Seddah, Farah Essaidi, Amal Fethi, Matthieu Futeral, Benjamin Muller, Pedro Javier Ortiz Suárez, Benoît Sagot, and Abhishek Srivastava. 2020. Building a user-generated content North-African Arabizi treebank: Tackling hell. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1139–1150, Online. Association for Computational Linguistics.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018a. Language identification and named entity recognition in Hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58, Melbourne, Australia. Association for Computational Linguistics.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018b. A Twitter corpus for Hindi-English code mixed POS tagging. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 12–17, Melbourne, Australia. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28, Portland, Oregon, USA. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Jeanine Treffers-Daller. 2020. Turkish-German code-switching patterns revisited: What naturalistic data can(not) tell us. *Advances in Contact Linguistics: In honour of Pieter Muysken*, 57:237.

Jeanine Treffers-Daller and Özlem Çetinoğlu. 2022. TuGeBiC: A Turkish German bilingual code-switching corpus. *arXiv*, abs/2205.00868.

Leonoor Van der Beek, Gosse Bouma, Rob Malouf, and Gertjan Van Noord. 2002. The Alpino dependency treebank. In *Computational linguistics in the Netherlands 2001*, pages 8–22. Brill Rodopi.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021. Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.

Emre Yılmaz, Maaike Andringa, Sigrid Kingma, Jelske Dijkstra, Frits van der Kuip, Hans Van de Velde, Frederik Kampstra, Jouke Algra, Henk van den Heuvel, and David van Leeuwen. 2016. A longitudinal bilingual Frisian-Dutch radio broadcast database designed for code-switching research. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4666–4669, Portorož, Slovenia. European Language Resources Association (ELRA).

Xiang Yu, Ngoc Thang Vu, and Jonas Kuhn. 2020. Ensemble self-training for low-resource languages: Grapheme-to-phoneme conversion and morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 70–78, Online. Association for Computational Linguistics.

## A Appendix

### A.1 Model Configuration and Hyper-parameters

We provide the configuration and hyper-parameters of the parser and sequence labeler models presented in Section 4.3.

$\texttt{Base}_{\texttt{LSTM}}$ We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.002, batch size of 16, and all dropout probabilities are set to 0.33 for the parser and the sequence labeler models. We train the parser for 150 epochs and sequence labeling tasks for 100 epochs.

We use 300-dimensional FastText embeddings (Grave et al., 2018) as pretrained word vectors. Since these embeddings are monolingual, we choose Russian FastText embeddings for Kpv-Ru, Hindi embeddings for Hi-En, Dutch embeddings for Fy-Nl, and Turkish embeddings for Tr-De treebanks. The model also uses 100-dimensional character embeddings and POS tag embeddings which are randomly initialized. The 3-layer BiLSTM modules of the parser and the sequence labeler have hidden layer size of 512 on each side. The decoder of the parser includes an arc MLP of size 512

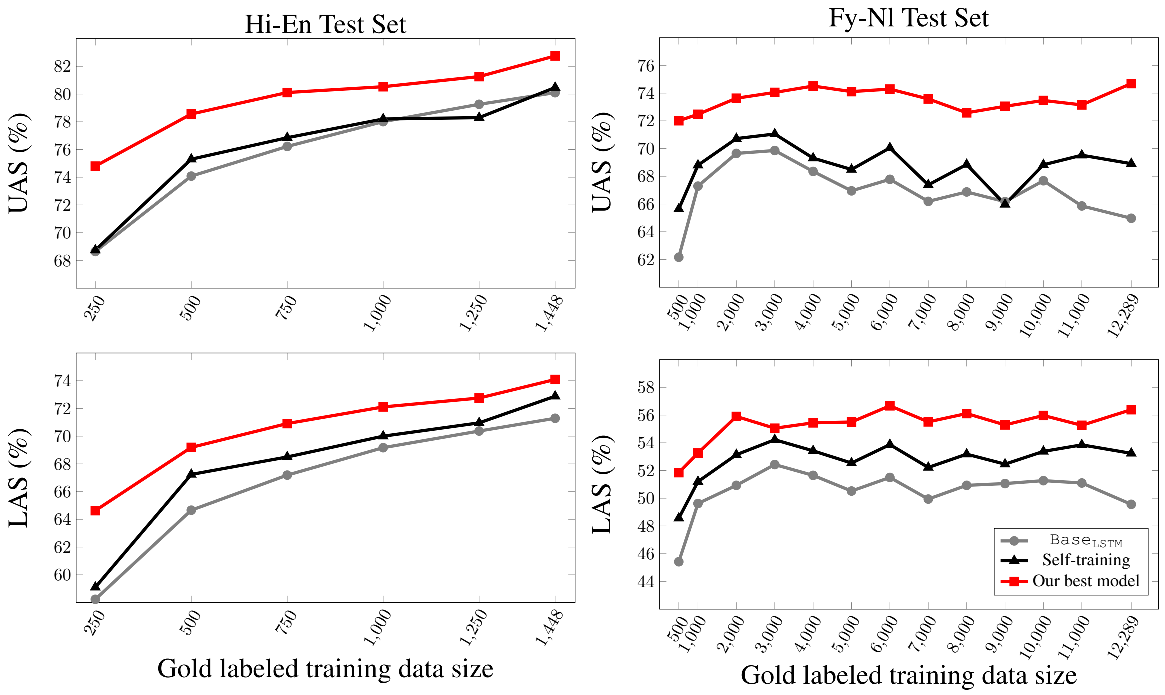


Figure 4: Comparison of $\texttt{Base}_{\text{LSTM}}$, `Self-training`, and our best model for Hi-En and Fy-Nl in terms of attachment scores.

and a label MLP of size 128. The decoder of the sequence labeler consists of two fully connected layers with size 128 and 64, respectively.

$\texttt{Base}_{\text{XLMR}}$ Due to computational efficiency, we choose the 768-dimensional XLM-R base language model as the word representation module of the $\texttt{Base}_{\text{XLMR}}$ architecture. For the parser, the arc MLP of the biaffine classifier has the same size with XLM-R model and the label MLP has the size of 256. Dropout for the classifier is set to 0.33. For the sequence labeler, we use a single-layer feed-forward neural network to extract logit vectors. We use AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 0.00004 and set batch size to 16. The number of epochs for the parser is 300 with an early stop of 50 epochs. For the sequence labeler, we train the models for 100 epochs with an early stop of 15 epochs.

### A.2 Effect of Gold Labeled Data on the Parsing Performance

In our main experiments the gold training data size differs among the four datasets. While the gold labeled data used for training of Kpv-Ru and Tr-De includes approximately 500 sentences, Hi-En has 1,448 gold labeled training CS data and for Fy-Nl we used the training set of the Dutch Alpino UD Treebank which consists of 12,289 gold labeled Dutch sentences. In order to observe how the amount of gold labeled training data affects the models' performance, we did a set of experiments on each of Hi-En and Fy-Nl datasets by incrementally increasing the size of labeled training data from 500 to the original training data size as used in the main experiments. Figure 4 shows results of these experiments.

We observe that our best model on these datasets (`+NOC,+DTR,+RPE` for Hi-En and `+NOC,+RPE,+LIH` for Fy-Nl) always surpasses `Self-training` and $\texttt{Base}_{\text{LSTM}}$ regardless of the available gold training data. Increasing the labeled training data has always a positive effect on the performance of all models for Hi-En but causes fluctuations in the performance for the case of Fy-Nl. The reason for this difference might be that the training data of Hi-En is in-domain and includes CS sentences, while the training data we use for Fy-Nl is out-of-domain and includes monolingual Dutch sentences.