A Language-aware Approach to Code-switched Morphological Tagging

Şaziye Betül Özateş and **Özlem Çetinoğlu**
Institute for Natural Language Processing, University of Stuttgart, Stuttgart, Germany
{saziye.oezates,ozlem.cetinoglu}@ims.uni-stuttgart.de

Abstract

Morphological tagging of code-switching (CS) data becomes more challenging especially when language pairs composing the CS data have different morphological representations. In this paper, we explore a number of ways of implementing a language-aware morphological tagging method and present our approach for integrating language IDs into a transformer-based framework for CS morphological tagging. We perform our set of experiments on the Turkish-German SAGT Treebank. Experimental results show that including language IDs to the learning model significantly improves accuracy over other approaches.

1 Introduction

Morphological tagging is a well known sequence labelling task in Natural Language Processing (NLP). It is the task of finding the correct morphological analysis for a given word form. The analysis is usually represented with a set of morphological features. Tagging these features is beneficial in solving most NLP tasks since having knowledge about the morphological analysis of natural language words gives clues about their syntactic nature and their roles in context (Müller and Schütze, 2015). Morphological tagging becomes more important when the language in question is a morphologically rich one and the part-of-speech (POS) information about word forms is not sufficient to syntactically classify them (Tsarfaty et al., 2013).

Morphological tagging is challenging in itself[1] and it becomes more challenging when the processed language is code-switched, a phenomenon that occurs when bilingual speakers frequently switch between languages and produce utterances

Form	POS	Morphological features
(a) In German:		
in	ADP	_
Autos	NOUN	Case=Dat | Gender=Neut | Number=Plur
(b) In Turkish:		
arabalarda	NOUN	Case=Loc | Number=Plur

Figure 1: The morphological analyses of German (a) and Turkish (b) translations of the phrase *in cars*.

that include word forms and phrases from both languages. The challenge amplifies as the linguistic difference between the composing languages increases. This is because unlike POS annotation that can be made common across languages (e.g. Universal Dependencies (Nivre et al., 2016)), morphological annotation is more language-specific. The example in Figure 1 shows this difference explicitly. Even though both *Autos* in German and *arabalarda* in Turkish share the same POS tag as NOUN, they have different morphological analyses. This difference stems from inherent properties of these languages. German employs grammatical gender while Turkish does not. Additionally in the example, the Turkish *locative* case corresponds to German *dative*. Such structural differences, combined with the rich morphology of individual languages taking part in CS data, make CS morphological tagging even more challenging with respect to CS POS tagging, a task that is a more common and more studied NLP task (cf. Section 2). In fact, there has not been any research focused on CS morphological tagging before.

We hypothesise that the language-dependent nature of morphological tagging can be solved more successfully for the case of CS data when the learning model has the knowledge of which language a word form belongs to. Starting from this hypothesis, we search ways of including the language ID (LID) information to tagging and present a language-aware approach. The proposed approach

[1] For instance, in the CoNLL 2018 Shared Task of Multilingual Parsing from Raw Text to Universal Dependencies, morphological tagging has the lowest range of scores among sentence segmentation, word segmentation, tokenisation, lemmatisation, and POS tagging. universaldependencies.org/conll18/results.html

Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, pages 72–83

https://doi.org/10.26615/978-954-452-056-4_010

integrates LIDs to the dense representation of input tokens in a transformer-based learning model. We conducted experiments on the only CS dataset with complete morphological annotation (Turkish-German SAGT Treebank (Çetinoğlu and Çöltekin, 2019)).[2] Results show that the proposed approach outperforms all of the baselines significantly and the use of LIDs is beneficial in tagging morphology for CS data. Our contributions are twofold: We present the first study on CS morphological tagging, and our data-driven method of integrating LIDs is applicable to any CS dataset and task that can exploit language IDs.

2 Related Work

Although there does not exist any prior study on CS morphological tagging, utilising language IDs in other CS tasks has been quite common. We divide how LID is utilised into three methods: as part of a pipeline, as part of joint processing, and as Machine Learning (ML) features. While one or more of these techniques have been applied to many CS tasks, e.g. parsing (Bhat et al., 2017), sentiment analysis (Vilares et al., 2016), and normalisation (van der Goot and Çetinoğlu, 2021), we focus here mainly on POS tagging, as it is a sequence labelling task and the closest one to morphological tagging.

One of the most commonly used pipeline approach is processing the data as monolingual fragments (Vyas et al., 2014; Jamatia et al., 2015; Barman et al., 2016; Bhat et al., 2017; AlGhamdi et al., 2016). For each language in the mixed data, a monolingual model is trained. During prediction, the input is split into fragments according to their language IDs and each fragment is processed by the respective monolingual model. The output is then merged into its original form. The advantage of this approach is to eliminate the need of CS data for training. However, context information is lost.

The other common pipeline approach is using LIDs in decision-making after getting predictions from monolingual models. In this setup the mixed input is given to both monolingual models. The predicted LID is then used to select the model output of the corresponding language. Solorio and Liu (2008) is the first to use this approach on English-Spanish POS tagging. Later Barman et al. (2016) and AlGhamdi et al. (2016) used this setup for English-Bengali-Hindi, and for English-Spanish and Modern Standard Arabic-Egyptian Arabic, as well as the first pipeline technique. While in Barman et al.'s (2016) case using the second pipeline method slightly outperforms the first one, AlGhamdi et al. (2016) show the first pipeline outperforms by a large margin. Thus we opted for the first architecture as one of our baselines.

Another model of Barman et al.'s (2016) was jointly trained LID and POS taggers that achieve a quite large improvement over their pipeline models. Soto and Hirschberg (2018) also trained LID and POS taggers together in their BiLSTM architecture. AlGhamdi and Diab (2019) choose joint LID and POS tagging as one of their architectures and show that distant language pairs Spanish-English and Hindi-English benefit from multi-task learning.

In many work from pre-neural era, LIDs are given as one of the features to ML models. While Solorio and Liu (2008) did not observe any significant improvement in doing so, Jamatia et al. (2015) shows that adding the LID of a token improves its POS tagging for English-Hindi. Sequiera et al. (2015) and Bhat et al. (2017) also inserted LID as a feature into their ML models. As a neural approach, Soto and Hirschberg (2018) represented the six LID labels existing in their data as boolean features and concatenated them with word vectors in a BiLSTM along with other features they used.

Different from the previous approaches, Aguilar and Solorio (2020) use language identification to create a code-switching ELMo from English ELMo (Peters et al., 2018). Later they show the effectiveness of their CS-ELMo by achieving state-of-the-art POS tagging results on a Hindi-English dataset (Singh et al., 2018). They also employ multi-task learning where their auxiliary task is language identification with a simplified LID tag set for LID, POS, and NER tagging.

3 Methodology

For morphological tagging of CS data, we chose to use STEPS[3] (Grünewald et al., 2020) as our framework. STEPS is an NLP tool for tagging and syntactic parsing in Universal Dependencies (UD) style (Nivre et al., 2016). Our motivation behind deciding on STEPS as our framework is based on two reasons. First, for token representation it utilises transformer-based language models, which

[2] There is also the NArabizi Treebank (Seddah et al., 2020) which includes partial morphological annotation where the total number of unique annotations is 46 in contrast to the SAGT Treebank which has 795 unique morphological annotations. Hence, we did not use this treebank in our study.

[3] github.com/boschresearch/steps-parser

have recently become famous for their outstanding success in various NLP tasks (Kondratyuk and Straka, 2019; Hoang et al., 2019). Second, STEPS is an open-source system with a minimum use of black-box modules that make the modification of the source codes very challenging, if not impossible. Moreover, STEPS is a current state-of-the-art NLP tool that outperformed other state-of-the-art tools Udify (Kondratyuk and Straka, 2019) and UDPipe 2.0 (Straka et al., 2019) in tagging and parsing of several languages (Grünewald et al., 2020).

Section 3.1 gives a brief description about STEPS. Sections 3.2 and 3.3 describe the baseline methods and our proposed approach for integrating LIDs to CS morphological tagging, respectively.

3.1 Framework

STEPS is mainly developed as a multilingual system for parsing. It also performs sequence labelling tasks such as POS and morphological tagging in a multi-task learning (MTL) setup. For our purposes, we adapted STEPS to solely perform sequence labelling. When this adapted version is used standalone, it becomes a baseline for our task. We mention this version as the `Standalone` approach throughout the paper.

The STEPS architecture follows Kondratyuk and Straka (2019) for computing token embeddings from the transformer-based language model and performing tagging and parsing. Token embeddings are calculated as a weighted sum of all intermediate outputs of the transformer layers. Coefficients of this weighted sum are learned during training. For sequence labelling, STEPS utilises a single-layer feed-forward neural network on top of token representations to extract the logit vectors for respective label vocabularies. More detailed information about the STEPS architecture can be found in (Grünewald et al., 2020).

3.2 Baselines for Language ID Integration

In a given dataset, the language-dependent morphological annotation of words that share the same POS tag gives us the intuition that feeding a model with token-wise LID information can help improve its accuracy for CS morphological tagging. Starting from this hypothesis, we designed and experimented with three ways of using token-level LID information in the model.

3.2.1 Data Split (`DSplit`)

One of the first methods that come to mind when dealing with CS data is splitting the data from CS points and treating the split parts as monolingual data as in the first pipeline method mentioned in Section 2. For our case, this method consists of three steps. First, input data is split to sub-parts containing monolingual data only. Second, monolingual models for each sub-part are trained. Each trained model processes its corresponding sub-part separately. In the last step, the output of models are joined to reach the processed version of the data.

To achieve the split of CS data into monolingual parts, we created a simple algorithm. Starting from the first token in a sentence, the algorithm creates sentence fragments whenever it encounters a switch between tokens with LIDs denoting one of the main languages in the CS data. Tokens with other LIDs (e.g., punctuation or mixed tokens where intra-word CS occurs) stay in the fragment created at that moment. Figure 2 depicts this process on a Turkish-German sentence.

3.2.2 Multi-Task Learning (`MTL`)

Another frequently applied method is the multi-task learning approach when two or more related tasks have the potential of benefitting each other through the domain information they contain. The main idea of this approach is improving the learning of a model for a task with the help of the knowledge contained by another task (Zhang and Yang, 2017). MTL has been shown effective in various areas in NLP (Collobert and Weston, 2008; Fang et al., 2019), especially in low-resource scenarios, usually as a way of transferring knowledge from a high-resource auxiliary task to a low-resource target task as in Lin et al. (2018). Our case is also a low-resource scenario where we have two related tasks, morphological tagging as the target and LID tagging as a simpler auxiliary task. In our setup, these two tasks are trained together with the same model and the loss is computed by summing losses of each task. The loss for LID tagging is scaled down 5% in training, as it was done for simpler tasks in (Grünewald et al., 2020). This loss scaling is for preventing the validation accuracy for LID tagging to go up too quickly and cause an underfitting for morphological tagging.

3.3 Proposal: LID Vectors (`LIDVec`)

Our proposal to integrate LIDs to the model is via creating LID embeddings and concatenating them

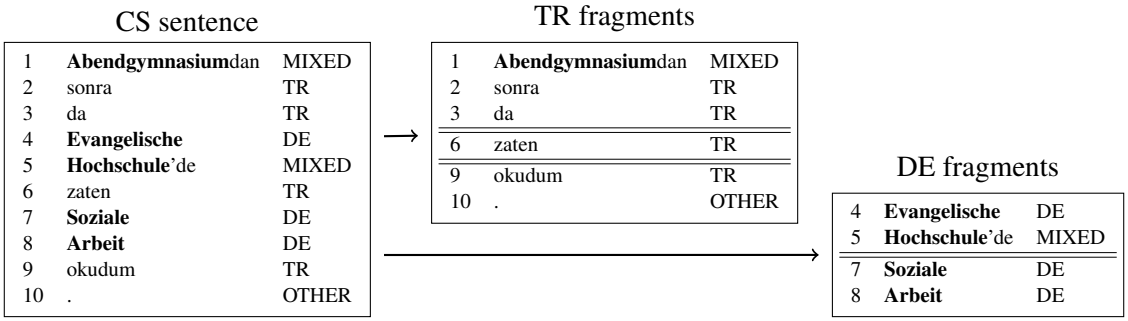

Figure 2: Splitting an example code-switching sentence to Turkish (TR) and German (DE) fragments. German tokens and token parts are shown in bold. (*Sentence translation: After the night school, I studied Social Work in the Protestant University.)*

to the embeddings of input tokens. The motivation behind this approach is to directly encode the LID information to each token inside the learning model and by this way to lessen the model's confusion caused by the tokens with different LIDs having different morphological annotations. Moreover, this way we can represent each LID label in contrast to `DSplit` that uses only main LID labels.

There are more than one method to represent LIDs as vectors inside the model. One-hot encoding of each LID is one of them.[4] Another method would be starting from a random embedding for each LID and training these embeddings with the rest of the model. Instead of random initialisation, LID embeddings can also be initialised with the average vectors of token embeddings in the training set, calculated for each LID label. Our motivation behind this clustering method is to see whether starting the training of the LID vectors from a more reasonable point will improve accuracy. We experimented with all of these models and chose to continue with the randomly initialised LID embeddings method based on our observation that this method works best among others. The comparison of these methods is discussed in Section 5.

In `LIDVec`, each LID label is assigned a 100-dimensional embedding vector at the beginning of training. The embedding of each input token is then concatenated with its corresponding LID embedding. These concatenated vectors are then given to the model for training. The loss at each epoch is backpropagated to both the token embeddings and the LID embeddings. We apply batch normalisation to token embeddings right after the concatenation.

[4] Soto and Hirschberg (2018) use a similar way. They represent LIDs as boolean features concatenated to word vectors in a BiLSTM architecture.

4 Experiments

4.1 Data

We evaluate our approaches on the Turkish-German SAGT Treebank (Çetinoğlu and Çöltekin, 2019) UD version 2.7.1.[5] It is based on a Turkish-German code-switching corpus created from conversation recordings of bilinguals. Although the treebank consists of spoken sentences, the transcriptions are normalised and hence the orthography does not pose a challenge in terms of morphological tagging. The SAGT Treebank includes five LID labels: `TR` for Turkish, `DE` for German, `LANG3` for tokens that belong to a third language other than Turkish and German, `OTHER` for punctuation, and `MIXED` for tokens with intra-word code-switching. Example (1) shows the structure of a mixed word from Figure 2.

(1) ***Abendgymnasium****dan*
night school.from
'from the night school'

Here the first part (*Abendgymnasium*) is a German noun and the second part (*-dan*) is a Turkish suffix. Although they are from different languages, the token ***Abendgymnasium****dan* has a single language ID since the two parts of the token are written orthographically together.

We use the original training, development, and test splits in experiments, only further splitting a small part from the development set as the fine-tuning set.[6] Sentence counts and LID distribution is given in Table 1. The average sentence length is 15.35 and the average code switches per sentence is

[5] `github.com/UniversalDependencies/UD_Turkish_German-SAGT/tree/dev`

[6] The fine-tuning set is created by randomly extracting equal amount of sentences from each document in the development set.

	Sent Count	Token Count					
		TR	DE	MIXED	LANG3	OTHER	**Total**
Tra	578	3,727	5,149	105	69	1,034	10,084
		37%	51%	1%	0.7%	10.3%	
FT	101	721	864	21	16	158	1,780
		41%	49%	1.2%	0.9%	8.9%	
Dev	700	4,389	5,589	122	48	1,128	11,276
		39%	49.6%	1%	0.4%	10%	
Test	805	5,341	7,139	183	46	1,384	14,093
		38%	50,6%	1.3%	0.3%	9.8%	
Total	2,184	14,178	18,741	431	179	3,704	37,233
		38%	50.3%	1.2%	0.5%	10%	

Table 1: Sentence and token counts of the Turkish-German SAGT Treebank used in the experiments (Tra: training, FT: fine-tuning, Dev: development).

	TR	DE	MIXED	LANG3	OTHER
Tags	526	293	53	5	1
Features	61	37	22	5	1

Table 2: The number of unique morphological tags and the number of unique morphological features for each language category in the SAGT Treebank.

2.19 on the whole treebank. The counts of unique morphological tags and morphological features that constitute the tags are depicted in Table 2.

Note that previous studies that follow a similar approach to DSplit use monolingual data that are usually available in large amounts in training (Vyas et al., 2014; Jamatia et al., 2015; Barman et al., 2016; Bhat et al., 2017; AlGhamdi et al., 2016). However we do not utilise monolingual Turkish and German data in the current setting of DSplit experiments. We experimented with using morphological features of two Turkish treebanks – IMST (Sulubacak et al., 2016) and BOUN (Türk et al., 2020) and two German treebanks – GSD (McDonald et al., 2013) and HDT (Borges Völker et al., 2019) as additional monolingual data but this resulted in a decrease in DSplit's accuracy possibly due to conflicting morphological annotations of these treebanks. So, we only use the corresponding parts of the SAGT Treebank in training and evaluation of DSplit. We also experimented with the second pipeline approach mentioned in Section 2. In line with our expectations, it gives worse performance. So, we stick to our current DSplit method (cf. Table 8 in Appendix A for a comparison of two approaches).

4.2 Model Configuration

STEPS can be used with both BERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020). We chose to use multilingual XLM-R observing it outperforms multilingual BERT in our preliminary experiments, which is in line with previous findings (Liang et al., 2020; Conneau et al., 2020). We use XLM-R$_{Base}$ with 12 layers and 768 hidden states in all the experiments. We stick to the default configuration of STEPS (Grünewald et al., 2020) for all the models except LIDVec. For LIDVec, token embedding size was changed from 768 to 868 since embeddings are expanded with the concatenation of 100-dimensional LID embeddings.

4.3 Predicted Language IDs

DSplit and LIDVec need LIDs; the former during splitting the dataset into languages, the latter during the concatenation of a token embedding with its corresponding LID vector. We evaluate these models with both gold and predicted LIDs. Predicted labels are obtained by training the STEPS Standalone model for LID tagging.

4.4 Metrics

We use accuracy as the evaluation metric. We count a morphological tag prediction of a token correct only when it is an exact match with the gold one. In addition to reporting the overall accuracy, we also provide accuracy on each LID label separately. This enables us to easily observe the parts each model has the most difficulty with. The significance between the performance of the models is measured using the randomisation test (van der Voet, 1994). When we mention a performance difference being *significant*, it means the difference is found statistically significant with $p < 0.05$.

4.5 Results

Table 3 shows experimental results for each model on the development and test sets.[7] It also demonstrates the evaluation of another baseline – Udify, a well-known, state-of-the-art transformer-based multi-task tool, which uses multilingual BERT as its language model (Kondratyuk and Straka, 2019).

We see that all three models that utilise LIDs outperform Standalone as well as Udify on both development and test sets. Although Standalone and Udify have similar architectures, the performance of the former surpasses that of the latter in terms of accuracy. Besides some design decisions, the main difference between these two models is the choice of the pretrained lan-

[7] The scores on the development set are the average of three separate runs while the scores on the test set are obtained by using the run that gives the best result in the development set.

Model	Accuracy on the Development Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
Udify (mBERT)	74.64	82.18	44.26	52.08	**99.91**	80.48
STEPS - Standalone	79.37	81.18	66.39	43.75	**99.91**	82.03
STEPS - MTL	79.94	82.05	71.04	43.75	**99.91**	82.74
STEPS - DSplit (w. gold LIDs)	81.17	83.03	69.67	**52.78**	**99.91**	83.72
STEPS - LIDVec (w. gold LIDs)	**81.87**	**83.54**	**73.22**	50.00	99.17	**84.20**
	Accuracy on the Test Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
Udify (mBERT)	71.95	79.21	39.34	34.78	99.86	77.83
STEPS - Standalone	76.15	77.15	61.75	47.83	**99.93**	78.71
STEPS - MTL	76.47	79.04	68.31	45.65	**99.93**	79.87
STEPS - DSplit (w. gold LIDs)	78.04	80.12	65.03	**50.00**	**99.93**	80.98
STEPS - LIDVec (w. gold LIDs)	**79.20**	**80.77**	**78.69**	34.78	98.77	**81.76**

Table 3: Morphological tagging accuracy of the models on the Turkish-German SAGT Treebank.

Model	Accuracy on the Development Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
STEPS - DSplit (w. pred. LIDs)	80.66	82.95	70.43	41.50	**100.0**	83.43
STEPS - LIDVec (w. pred. LIDs)	**81.85**	**83.53**	**73.22**	**49.31**	99.17	**84.18**
	Accuracy on the Test Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
STEPS - DSplit (w. pred. LIDs)	77.65	80.01	65.59	**48.78**	**100.0**	80.78
STEPS - LIDVec (w. pred. LIDs)	**79.22**	**80.73**	**78.69**	34.78	98.77	**81.75**

Table 4: Morphological tagging accuracy of DSplit and LIDVec when predicted LID labels were used.

	Accuracy	
	Development set	**Test set**
TR	99.09	99.42
DE	98.43	98.80
MIXED	90.16	92.90
LANG3	52.08	67.39
OTHER	99.91	99.86
ALL	98.55	98.96

Table 5: LID prediction accuracy of STEPS on the development and test sets of the SAGT Treebank.

guage model. While Udify uses multilingual BERT, Standalone utilises XLM-R.

The best performing model is LIDVec as we expected. It outperforms Standalone more than 2 and 3 points on the development and test sets, respectively. The two baselines for LID integration, DSplit and MTL, perform better than Standalone although they are less successful than LIDVec. We observe that integrating LIDs to the system improves the accuracy in morphological tagging in all three scenarios, although the amount of the improvement differs across the models.

To see how LID prediction affects DSplit and LIDVec, we repeated the same experiments with predicted LIDs. The results are given in Table 4. As introduced in Section 4.3, Standalone is used for LID tagging. Its performance on the development and test sets is shown in Table 5.

In Table 4, we see that LID accuracy has a stronger influence on DSplit while LIDVec stays almost unaffected. This might stem from LIDs playing a key role in DSplit by splitting the data into monolingual parts that are then used to train two separate models. So, the errors in LIDs are more explicitly propagated to the two models that learn to predict the morphological features of monolingual data only. However, LIDs have a more implicit effect in LIDVec. The errors in LIDs cause the wrong LID vector to be concatenated to the embeddings of some tokens but this error can later be compensated through the training of the whole model where both token and LID embeddings being updated at each step. Considering the high overall accuracy in LID prediction in Table 5, LIDVec seems to compensate the small error rate in predicted LIDs. Although LANG3 prediction accuracy is low, this does not cause a substantial effect in the overall accuracy of LID prediction since this label is rare in the treebank.

5 Analysis on LID Integration

LID representation and initialisation In Section 3.3, we mention two more ways in addition to our preferred approach for the representation of LIDs as vectors. The first way is representing LIDs as one-hot vectors. We define each LID label as a one-hot vector and concatenate these vectors with token embeddings provided by the lan-

guage model as in LIDVec. We experimented with this approach on the development set. However, this method showed poorer performance than Standalone which does not utilise LIDs in any way. We believe that one-hot vector representation might be too rigid to be used together with token embeddings due to the fact that the range of the values in these two representations greatly vary.

The second method for the LID vector representation includes the initialisation of LID embeddings by averaging the embeddings of same-LID tokens in the training set. In the initial experiments we see that when we use the average initialisation instead of a random initialisation, the training phase progresses faster and the learning stops early when the training accuracy is around 85%, in contrast to the random initialisation in which the training phase ends after a higher number of epochs and with a higher training accuracy. So, we extended the training time by changing the early stop criteria from 15 epochs to 50 epochs to give the average initialisation an opportunity to show its true capacity. Figure 3 compares the performance of these two initialisation methods for two different early stop criteria on the development set. We see that the underfitting in the average initialisation method is eliminated as the number of epochs increases. Overall, the performance of both initialisation methods is the same when they are trained sufficiently. We conclude that random initialisation can be preferred if there are time restrictions.

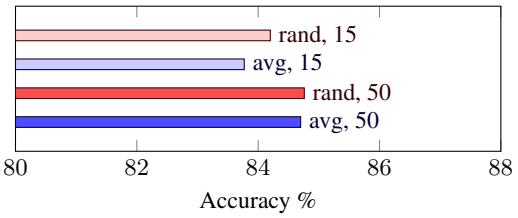

Figure 3: Comparison of random vs. average initialisation in the LIDVec model when the early stop criteria is 15 epochs vs. 50 epochs.

The impact of LID prediction We proposed three different approaches for LID integration. In terms of resources needed, MTL does not need an external LID prediction by definition, since it predicts LIDs and morphology jointly. However, it is also the worst performing one among the three approaches. DSplit and LIDVec both outperform MTL, but require predicted LIDs to function.

To test how sensitive these models to the LID prediction accuracy, we evaluated DSplit and LIDVec with MarMoT, a CRF-based sequence tagger (Müller et al., 2013) which has ~96% accuracy in LID prediction instead of the STEPS LID model with ~99% accuracy (cf. Table 9 in Appendix B for complete results). Although LIDVec's performance stays almost unaffected by the accuracy drop in LID prediction, DSplit accuracy drops approximately 1 point and more than 2 points in development and test sets, respectively. We conclude that DSplit is more vulnerable to LID accuracy whereas LIDVec can be paired with a faster and computationally less costly LID model if needed be. Another disadvantage of DSplit is the need to train multiple monolingual models to deal with different languages in CS data, in contrast to the single model architecture of LIDVec. DSplit also requires pre- and post-processing of the input and output, respectively. Considering its superior performance, and the robustness and compactness of its architecture, we suggest LIDVec as the best approach to CS morphological tagging among the models discussed in this paper.

The impact of LIDs on POS tagging We also performed experiments for POS tagging, the other possible sequence labelling task we can employ LID integration. Table 6 shows the overall accuracies for each model on the development and test sets of the SAGT Treebank. We do not observe any significant difference between the accuracies of the models, which is in line with our expectations. This is because universal POS tags used in the SAGT treebank are common to all languages in contrast to morphological tags that include many language-specific features. Hence, identifying the language a token belongs to does not add extra benefits in POS prediction.

Model	Accuracy	
	Dev	Test
STEPS - Standalone	93.72	**92.27**
STEPS - MTL	93.74	92.10
STEPS - DSplit (w. gold LIDs)	93.53	92.07
STEPS - LIDVec (w. gold LIDs)	**93.94**	92.24

Table 6: The POS tagging accuracy scores of the models on the development and test sets of the Turkish-German SAGT Treebank.

6 Qualitative Analysis

Most Common Improvements We observe that integrating language IDs contributes to a 10% in-

crease in predicting the presence of possessive markers in Turkish nouns, which are not a feature of German nouns. This is something expected since providing LIDs enables the model to differentiate between the different sets of morphological features of two languages better. Similarly, the LID knowledge makes a 4% enhancement in predicting the existence of the *Gender* feature that is present in German nouns but absent in Turkish ones (cf. Figure 1). To understand this better, we compared LIDVec and Standalone in terms of their feature-based success. In this feature-based performance measurement, partial matches are also given scores in contrast to the evaluation metric we adopted, which counts a predicted morphological tag as correct only if it is an exact match – i.e., all the features that constitute the morphological tag are predicted correctly. We measure the feature-based performance of the models by dividing each morphological tag into features and counting each feature match as a point. Table 7 compares feature-based results of LIDVec and Standalone. We observe that LIDVec improves both *Precision* and *Recall* by more than 2%. These results suggest that LIDVec facilitates predicting the full set of features.

Model	Precision	Recall	F1	Acc
Standalone	87.72	87.19	87.24	82.03
LIDVec	**89.96**	**89.41**	**89.50**	**84.20**

Table 7: Feature-based partial scores of Standalone and LIDVec models on the development set of the Turkish-German SAGT Treebank.

When we look at what categories benefit most from including LIDs, we see that for Turkish they are verbs and nouns with an improvement of 11% and 10%, respectively. For German they are pronouns and nouns with 9% improvement. The success of morphology prediction for German verbs is already high for all models. Hence, there is not much improvement in German verbs. We observe that all the nouns and pronouns in both languages and also the verbal nouns in Turkish which are derived from verbs have the *Case* feature in their morphological analyses.

Confusion in `Case` feature values Although all models easily predicted the existence of the *Case* feature, they had the most trouble in deciding the value of it. Hence, we created confusion matrices of Standalone and LIDVec for different values of the *Case* feature on the development set as given in Figure 4. There are only four case markers in German: *nominative*, *accusative*, *dative*, and *genitive*. In Turkish, there are three additional case markers, namely *ablative*, *instrumental*, and *locative*. Albeit having a German lemma, MIXED tokens in the SAGT Treebank are annotated according to Turkish morphological annotation style due to the presence of Turkish suffixes in them. We observe that the most confusion occurs between *nominative* and *accusative* cases for all three token types. This confusion in TR and MIXED tokens results from the fact that the accusative suffix which makes the case of a word *accusative* and the possessive suffix in *nominative* nouns sometimes correspond to the same form in Turkish. In DE tokens, the situation is similar in the sense that *nominative* and *accusative* forms of German articles are different only for masculine, whereas they have the same form when their gender is feminine or neutral, or when they are in plural. LIDVec consistently reduces this confusion and predicts correct cases that plays an important role in its overall performance.

Improvement on `MIXED` tokens When observing the results in Tables 3 and 4, the notable success of LIDVec on predicting morphological analyses of MIXED tokens caught our attention. Even when predicted LIDs are used, LIDVec outperforms Standalone by a large margin in the development and test sets. We observe that MIXED tokens in the SAGT Treebank are mostly *nouns*. Therefore MIXED tokens get their share from overall *Case* improvements. When proportioned to the total number of cases in each category, the success of LIDVec is most visible in MIXED tokens.

Performance of `LIDVec` on `LANG3` and `OTHER` tokens We observe a pattern in the results that seems like a trade-off between the success on TR, DE, and MIXED and the success on LANG3 and OTHER. This is most visible in LIDVec. We do not see the consistent improvement trend over Standalone in LANG3 and OTHER accuracies as in TR, DE, and MIXED accuracies. To inspect this case, we compare confusion matrices of Standalone and LIDVec in Figure 5 for LANG3 and OTHER types. Both models confused LANG3 mostly with DE. We believe this situation stems from the fact that LANG3 tokens in the treebank are mostly English proper nouns and some of them are also common in German. Nonetheless, the low success rates in this token type by all

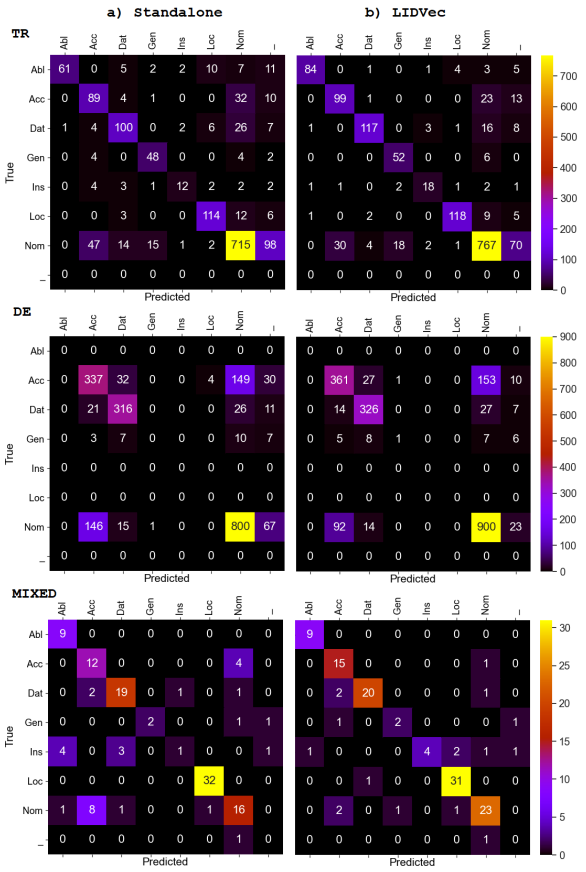

Figure 4: Confusion matrices of `Standalone` and `LIDVec` for different *Case* values.

models demonstrate once again how important the amount of training data is for data-driven models.

On the contrary, all models perform very well in predicting the absence of morphology in `OTHER` tokens. However, `LIDVec` makes a few more false predictions than `Standalone`. We believe this might stem from a slight overfitting of `LIDVec` towards `TR` tokens. Yet, accuracy of all models are above 98% for this type and we need more data to justify that there is a difference between the models for morphology prediction of `OTHER` tokens.

7 Conclusion

In this paper, we tackle the morphological tagging problem for CS data. We present some challenging aspects of the task and suggest the use of token-wise LID information. We experience with different ways of using LIDs on a transformer-based model and propose the LID Vectors approach. Our proposed model outperforms all the baselines significantly and proves to be a robust and compact way of LID integration. Being first on focusing morphological tagging on CS data,

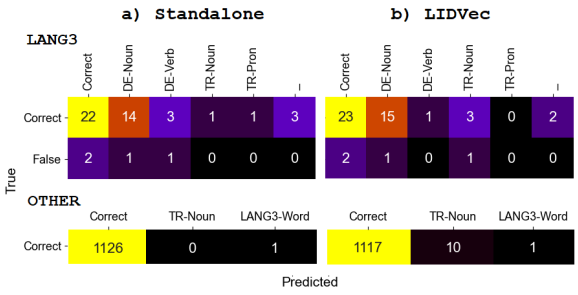

Figure 5: Confusion matrices for the tokens with `LANG3` and `OTHER` LID labels on the development set.

our study shows that utilising LIDs is an effective method in this task. We also give the first results on LID, POS, and morphological tagging on the Turkish-German SAGT dataset. An implementation of our model is available at `https://github.com/sb-b/steps-parser`.

Acknowledgements

We thank Stefan Grünewald for his valuable help in using the STEPS tool. This work is funded by DFG via project CE 326/1-1 "Computational Structural Analysis of German-Turkish Code-Switching" (SAGT).

References

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.

Fahad AlGhamdi and Mona Diab. 2019. Leveraging pretrained word embeddings for part-of-speech tagging of code switching data. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 99–109, Ann Arbor, Michigan. Association for Computational Linguistics.

Fahad AlGhamdi, Giovanni Molina, Mona Diab, Thamar Solorio, Abdelati Hawwari, Victor Soto, and Julia Hirschberg. 2016. Part of speech tagging for code switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 98–107, Austin, Texas. Association for Computational Linguistics.

Utsab Barman, Joachim Wagner, and Jennifer Foster. 2016. Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modelling. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 30–39, Austin, Texas. Association for Computational Linguistics.

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.

Emanuel Borges Völker, Maximilian Wendt, Felix Hennig, and Arne Köhn. 2019. HDT-UD: A very large Universal Dependencies treebank for German. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 46–57, Paris, France. 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.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

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.

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.

Wei Fang, Moin Nadeem, Mitra Mohtarami, and James Glass. 2019. Neural multi-task learning for stance prediction. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 13–19, Hong Kong, China. Association for Computational Linguistics.

Stefan Grünewald, Annemarie Friedrich, and Jonas Kuhn. 2020. Graph-based universal dependency parsing in the age of the transformer: What works, and what doesn't. *arXiv preprint arXiv:2010.12699*.

Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. 2019. Aspect-based sentiment analysis using BERT. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 187–196, Turku, Finland. Linköping University Electronic Press.

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.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. XGLUE: A new benchmark dataset for cross-lingual pretraining, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018, Online. Association for Computational Linguistics.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 799–809, Melbourne, Australia. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 526–536, Denver, Colorado. 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).

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. 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.

Royal Sequiera, Monojit Choudhury, and Kalika Bali. 2015. POS tagging of Hindi-English code mixed text from social media: Some machine learning experiments. In *Proceedings of the 12th International Conference on Natural Language Processing*, pages 237–246, Trivandrum, India. NLP Association of India.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018. 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.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for English-Spanish code-switched text. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1060, Honolulu, Hawaii. Association for Computational Linguistics.

Victor Soto and Julia Hirschberg. 2018. Joint part-of-speech and language ID tagging for code-switched data. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 1–10, Melbourne, Australia. Association for Computational Linguistics.

Milan Straka, Jana Straková, and Jan Hajič. 2019. Evaluating contextualized embeddings on 54 languages in POS tagging, lemmatization and dependency parsing. *arXiv preprint arXiv:1908.07448*.

Umut Sulubacak, Memduh Gökırmak, Francis Tyers, Çağrı Çöltekin, Joakim Nivre, and Gülşen Eryiğit. 2016. Universal Dependencies for Turkish. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3444–3454, Osaka, Japan. The COLING 2016 Organizing Committee.

Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. 2013. Parsing morphologically rich languages: Introduction to the special issue. *Computational Linguistics*, 39(1):15–22.

Utku Türk, Furkan Atmaca, Şaziye Betül Özateş, Gözde Berk, Seyyit Talha Bedir, Abdullatif Köksal, Balkız Öztürk Başaran, Tunga Güngör, and Arzucan Özgür. 2020. Resources for Turkish dependency parsing: Introducing the BOUN treebank and the BoAT annotation tool. *arXiv preprint arXiv:2002.10416*.

Rob van der Goot and Özlem Çetinoğlu. 2021. Lexical normalization for code-switched data and its effect on POS tagging. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics.

Hilko van der Voet. 1994. Comparing the predictive accuracy of models using a simple randomization test. *Chemometrics and Intelligent Laboratory Systems*, 25(2):313–323.

David Vilares, Miguel A. Alonso, and Carlos Gómez-Rodríguez. 2016. EN-ES-CS: An English-Spanish code-switching twitter corpus for multilingual sentiment analysis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4149–4153, Portorož, Slovenia. European Language Resources Association (ELRA).

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS tagging of English-Hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979, Doha, Qatar. Association for Computational Linguistics.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

A Comparison of Two Approaches for the Data Split Method

Model	Accuracy on the Development Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
first	**81.17**	**83.03**	**69.67**	52.78	**99.91**	**83.72**
second	81.05	82.78	60.93	**57.64**	99.88	83.48
	Accuracy on the Test Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
first	**78.04**	**80.12**	**65.03**	50.00	**99.93**	**80.98**
second	77.44	79.59	60.11	50.00	99.86	80.42

Model	Accuracy on the Development Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
first	**80.66**	**82.95**	**70.43**	41.50	**100.0**	**83.43**
second	80.50	82.81	63.44	41.50	99.97	83.23
	Accuracy on the Test Set					
	TR	DE	MIXED	LANG3	OTHER	**ALL**
first	**77.65**	**80.01**	**65.59**	48.78	100.0	**80.78**
second	77.05	79.54	60.75	48.78	100.0	80.26

Table 8: Morphological tagging accuracy of the two pipeline approaches for the DSplit method. The first part shows the scores in the existence of gold LIDs and the second part demonstrates the results when predicted LIDs are used instead of gold ones.

B Comparison of MarMoT and STEPS for LID Prediction

	Accuracy			
	Development set		Test set	
	MarMoT	**STEPS**	**MarMoT**	**STEPS**
TR	96.40	**99.09**	97.38	**99.42**
DE	97.84	**98.43**	97.88	**98.80**
MIXED	23.77	**90.16**	27.32	**92.90**
LANG3	41.67	**52.08**	0.0	**67.39**
OTHER	98.23	**99.91**	99.06	**99.86**
ALL	96.12	**98.55**	96.57	**98.96**

Table 9: Comparsion of MarMoT and STEPS tools for LID prediction on the development and test sets of the SAGT Treebank.