Multilingual Code-Switching for Zero-Shot Cross-Lingual Intent Prediction and Slot Filling

Jitin Krishnan **Antonios Anastasopoulos** **Hemant Purohit** **Huzefa Rangwala**
George Mason University
Fairfax, VA, USA
{jkrishn2,antonis,hpurohit,rangwala}@gmu.edu

Abstract

Predicting user intent and detecting the corresponding slots from text are two key problems in Natural Language Understanding (NLU). Since annotated datasets are only available for a handful of languages, our work focuses particularly on a zero-shot scenario where the target language is *unseen* during training. In the context of zero-shot learning, this task is typically approached using representations from pre-trained multilingual language models such as mBERT or by fine-tuning on data automatically translated into the target language. We propose a novel method which augments monolingual source data using multilingual code-switching via random translations, to enhance generalizability of large multilingual language models when *fine-tuning* them for downstream tasks. Experiments on the MultiATIS++ benchmark show that our method leads to an average improvement of $+4.2\%$ in accuracy for the intent task and $+1.8\%$ in F1 for the slot-filling task over the state-of-the-art across 8 typologically diverse languages. We also study the impact of code-switching into different families of languages on downstream performance. Furthermore, we present an application of our method for crisis informatics using a new human-annotated tweet dataset of slot filling in English and Haitian Creole, collected during the Haiti earthquake.[1]

1 Introduction

A cross-lingual setting is typically described as a scenario in which a model trained for a particular task in one *source* language (e.g. English) should be able to generalize well to a different *target* language (e.g. Japanese). While semi-supervised solutions (Muis et al., 2018; FitzGerald, 2020, *inter alia*) assume some target language data or translators are available, a zero-shot solution (Eriguchi et al., 2018; Srivastava et al., 2018; Xu et al., 2020) assumes none is available at training time. Having models that generalize well even to unseen languages is crucial for tackling real world problems such as extracting relevant information during a new disaster (Nguyen et al., 2017; Krishnan et al., 2020) or detecting hate speech (Pamungkas and Patti, 2019; Stappen et al., 2020), where the target language might be of low-resource or unknown.

Intent prediction and slot filling are two NLU tasks, usually solved jointly, which learn to model the intent (sentence-level) and slot (word-level) labels. Such models are currently used extensively for goal-oriented dialogue systems, such as Amazon's Alexa, Apple's Siri, Google Assistant, and Microsoft's Cortana. Finding the '*intent*' behind the user's query and identifying relevant '*slots*' in the sentence to engage in a dialogue are essential for effective conversational assistance. For example, users might want to '*play music*' given the slot labels '*year*' and '*artist*' (Coucke et al., 2018), or they may want to '*book a flight*' given the '*airport*' and '*locations*' slot labels (Price, 1990). A strong correlation between the two tasks has made jointly trained models successful (Goo et al., 2018; Haihong et al., 2019; Hardalov et al., 2020; Chen et al., 2019). In a cross-lingual setting, the model should be able to learn this joint task in one language and transfer knowledge to another (Upadhyay et al., 2018; Schuster et al., 2019; Xu et al., 2020). This is the premise of our work.

Highly effective transformer-based multilingual models such as mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020a) have found success across several multilingual tasks in recent years. In the zero-shot cross-lingual transfer setting with an unknown target language, a typical solution is to use pre-trained transformer models and fine-tune to the downstream task using the monolingual source

[1] To appear at Multiliingual Representation Learning Workshop at EMNLP 2021. Implementation and dataset are available at https://github.com/jitinkrishnan/Multilingual-ZeroShot-SlotFilling.

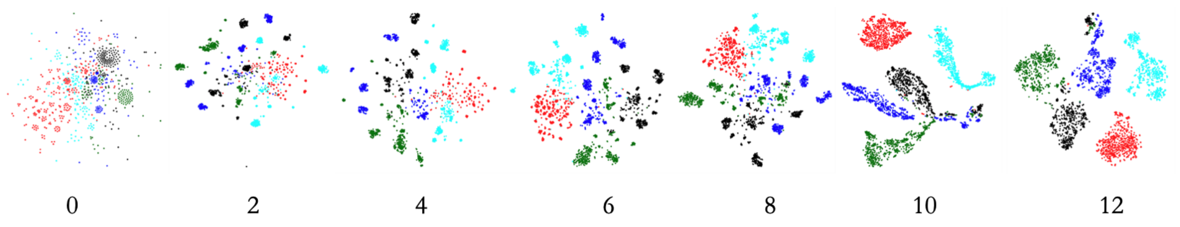


Figure 1: t-SNE plot of embeddings across the 12 multi-head attention layers of multilingual BERT. Parallelly translated sentences of MutiATIS++ dataset are still clustered according to the languages: English (black), Chinese (cyan), French (blue), German (green), and Japanese (red).

Intent	atis_airfare									
	Original (English)									
Words	Show	me	round	trip	fares	from	Denver	to	Philadelphia	
Slots	O	O	B-round_trip	I-round_trip	O	O	B-fromloc.city_name	O	B-fromloc.city_name	
	Chunk-level Code-Switched									
Words	给	我	看	看	ਸੈਰ	tarifas	desde	Denver	إلى	Филадельфия
Slots	O	O	O	O	B-round_trip	O	O	B-fromloc.city_name	O	B-fromloc.city_name

Figure 2: An original example in English from MultiATIS++ dataset and its multilingually code-switched version. In the above code-switching example, the chunks are in Chinese, Punjabi, Spanish, English, Arabic, and Russian. '*atis_airfare*' represents an intent class where the user seeks price of a ticket.

data (Xu et al., 2020). However, Pires et al. (2019) showed that existing transformer-based representations may exhibit systematic deficiencies for certain language pairs. Figure 1 also verifies that the representations across the 12 multi-head attention layers of mBERT are still not shared across languages, instead forming clearly distinguishable clusters per language. This leads to a fundamental challenge that we address in this work: enhancing the language neutrality so that the fine-tuned model is generalizable across languages for the downstream task. To this goal, we introduce a data augmentation method via multilingual code-switching, where the original sentence in English is code-switched into randomly selected languages. For example, chunk-level code-switching creates sentences with phrases in multiple languages as shown in Figure 2. We show that mBERT can be fine-tuned for many languages starting only with monolingual source-language data, leading to better performance in zero-shot settings.

Further, we show how code-switching with languages from different language families impacts the model's performance on individual target languages, even finding some counter-intuitive results. For instance, training on data code-switched between English and Sino-Tibetan languages is as helpful for Hindi (an Indo-Aryan Indo-European language) as code-switching with other Indo-Aryan languages, and Turkic languages can be helpful for both Chinese and Japanese.

Contributions: **a)** We present a data augmentation method via multilingual code-switching to enhance the language neutrality of transformer-based language models such as mBERT for fine-tuning to a downstream NLU task of intent prediction and slot filling. **b)** By studying different language families, we show how code-switching can be used to aid zero-shot cross-lingual learning for low-resource languages. **c)** We release a new human-annotated tweet dataset, collected during Haiti earthquake disaster, for intent prediction and slot filling in English and Haitian Creole.

2 Methodology

This section describes our problem definition, code-switching algorithm, language families, and the training methodology.

2.1 Problem Definition

Given a source (S) and a set of target (T) languages, the goal is to train a classifier using data only in the source language and predict examples from the completely unseen target languages. We assume the target language is *unknown* during training (fine-tuning) time, which makes direct translation to target infeasible. In this context, we use

Algorithm 1: Data Augmentation via Multilingual Code-Switching (Chunk-Level)

```
Input: $X_{ut}^{en}$, $y^{en}$, $y_{sl}^{en}$, $l_T$
Output: $X_{ut}^{cs}$, $y^{cs}$, $y_{sl}^{cs}$
$X_{ut}^{cs} \leftarrow \emptyset$, $y^{cs} \leftarrow \emptyset$, $y_{sl}^{cs} \leftarrow \emptyset$
$lset = googletrans.languages - l_T$
for $i \in 1..\ k$ do
    for $j \in 1..\ len(X_{ut}^{en})$ do
        $G^{cs} \leftarrow \emptyset$, $L^{cs} \leftarrow \emptyset$
        $chunks = slot\_chunks(X_{ut}^{en}[j],\ y_{sl}^{en}[j])$
        for $c \in chunks$ do
            $l \leftarrow random.choice(lset)$
            $t \leftarrow translate(c, l)$
            $G^{cs} \leftarrow G^{cs} \cup t$
            $L^{cs} \leftarrow L^{cs} \cup align\_label(c, t)$
        end
        $X_{ut}^{cs} \leftarrow X_{ut}^{cs}\ \cup\ G^{cs}$
        $y^{cs} \leftarrow y^{cs}\ \cup\ y^{cs}[j]$
        $y_{sl}^{cs} \leftarrow y_{sl}^{cs}\ \cup\ L^{cs}$
    end
end
```

code-switching (cs) to augment the monolingual source data. Thus, the input, augmented input, and output of our problem can be defined as:

Input: X_{ut}^{S}, y^{S}, y_{sl}^{S}, l_T
Code-Switched Input: X_{ut}^{cs}, y^{cs}, y_{sl}^{cs}
Output: y^{T}, $y_{sl}^{T} \leftarrow predict(X_{ut}^{T})$

where X_{ut} represents sentences, y their ground truth intent classes, y_{sl} the slot labels for the words in those sentences, and l_T the set of target languages. An example sentence, its intent class, and slot labels are shown in Figure 2.

2.2 Multilingual Code-Switching

Multilingual masked language models, such as mBERT (Devlin et al., 2019), are trained using large datasets of publicly available unlabeled corpora such as Wikipedia. Such corpora largely remain monolingual at the sentence level because the presence of intra-sentence code-switched data in written texts is likely scarce. The masked words that needed to be predicted usually are in the same language as their surrounding words. We study how code-switching can enhance the language neutrality of such language models by augmenting it with artificially code-switched data for fine-tuning it to a downstream task. Algorithm 1 explains this code-switching process at the chunk-level. When using slot filling datasets, slot labels that are grouped by BIO (Ramshaw and Marcus, 1999) tags constitute natural chunks, as shown in Figure 2. To summarize the algorithm, we take a sentence, take each chunk from that sentence, perform a translation into a random language using Google's NMT system (Wu et al., 2016), and align the slot labels to fit the translation, i.e., label propagation through alignment as the translated sentence do not preserve the number and order of words in the original sentence. At the chunk-level, we use a direct alignment. The BIO-tagged labels are recreated for the translated phrase based on the word tokens. More complex methods could be applied here to improve the alignment of the slot labels such as fast-align (Dyer et al., 2013) or soft-align (Xu et al., 2020), but we leave this for future work. Code-Switching at the word-level essentially translates every word randomly, while at the sentence-level translates the entire sentence. During the experimental evaluation process, to build a language-neutral model using monolingual source (English) data, **all eight target languages are excluded** from the code-switching procedure to avoid unfair model comparisons, i.e. removing target languages (l_T) from $lset$ in Algorithm 1.

Group Name	Languages
Afro-Asiatic	Arabic (ar), Amharic (am), Hebrew (he), Somali (so)
Germanic	German (de), Dutch (nl), Danish (da), Swedish (sv), Norwegian (no)
Indo-Aryan	Hindi (hi), Bengali (bn), Marathi (mr), Nepali (ne), Gujarati (gu), Punjabi (pa)
Romance	Spanish (es), Portuguese (pt), French (fr), Italian (it), Romanian (ro)
Sino-Tibetan, Koreanic, & Japonic	Chinese (zh-cn), Japanese (ja), Korean (ko)
Turkic	Turkish (tr), Azerbaijani (az), Uyghur (ug), Kazakh (kk)

Table 1: Selected language families to evaluate their impact on a target language.

Complexity. The augmentation process is repeated k times per sentence producing a new augmented dataset of size $k \times n$, where n is the size of the original dataset, i.e. space complexity of $\mathcal{O}(k \times n)$. For T translations per sentence, Algorithm 1 has a runtime complexity of $\mathcal{O}(k \times n \times T)$ assuming constant time for alignment. Word-level requires as many translations as the number of words but sentence-level requires only one. An increase in the dataset size also increases the training time, but the advantage is one model appropriate for many languages.

2.3 Language Families

A language family is defined as a group of related languages that likely share a common ancestor. For example, Portuguese, Spanish, French, Italian, and Romanian are all derived from Latin (Rowe and Levine, 2017). We use language families to study their impact on the target languages. We augment the source language with code-switching to a particular language family. For instance, code-switching the English dataset with Turkic language family and testing on Japanese can reveal how closely the two are aligned in the vector space of a pre-trained multilingual model. We work with 6 language groups: Afro-Asiatic (Voegelin and Voegelin, 1976), Germanic (Harbert, 2006), Indo-Aryan (Masica, 1993), Romance (Elcock and Green, 1960), and Turkic (Johanson and Johanson, 2015), also grouping Sino-Tibetan, Koreanic and Japonic (Shafer, 1955; Miller, 1967).[2] Germanic, Romance, and Indo-Aryan are genera of the Indo-European family. Language groups and corresponding languages are shown in Table 1. Each group is selected based on a target language in the dataset, and the Afro-Asiatic family is added as an extra group. In experiments, $lset$ in Algorithm 1 will be assigned languages from a specific family.

2.4 Joint Training

Joint training is traditionally used for intent prediction and slot filling to exploit the correlation between the two tasks. This is done by feeding the feature vectors of one model to another or by sharing layers of a neural network followed by training the tasks together. So, a standard joint model loss can be defined as a combination of intent (L_i) and slot (L_{sl}) losses. i.e., $L = \alpha L_i + \beta L_{sl}$, where α and β are corresponding task weights. Prior works (Goo et al., 2018; Schuster et al., 2019; Liu and Lane, 2016; Haihong et al., 2019) that use BiLSTM or RNN are now modified to BERT-based implementations explored in more recent works (Chen et al., 2019; Hardalov et al., 2020; Xu et al., 2020). A standard $Joint$ model consists of BERT outputs from the final hidden state (classification (CLS) token for intent and m word tokens for slots) fed to linear layers to get intent and slot predictions. Assuming h_{cls} represents the CLS token and h_m represents a token from the remaining word-level tokens, the BERT model outputs are defined as (Chen et al., 2019; Xu et al., 2020):

$$\begin{aligned} p^i &= softmax(W^i h_{cls} + b^i) \\ p^{sl}_m &= softmax(W^{sl} h_m + b^{sl}) \ \forall m \end{aligned} \tag{1}$$

with a multi-class cross-entropy loss[3] for both intent (L_i) and slots (L_{sl}). We will use this model as our baseline for joint training. Our goal will be to show that code-switching on top of joint training improves the performance. The output of Algorithm 1 will be the input used for joint training on BERT for code-switched experiments.

3 Datasets

Benchmark Dataset. We use the latest multilingual benchmark dataset of MultiATIS++ (Xu et al., 2020), which was created by manually translating the original ATIS (Price, 1990) dataset from English (en) to 8 other languages: Spanish (es), Portuguese (pt), German (de), French (fr), Chinese (zh), Japanese (ja), Hindi (hi), and Turkish (tr). The dataset consists of utterances for each language with an *'intent'* label for *'flight intent'* and *'slot'* labels for the word tokens in BIO format. A sample datapoint in English is shown in Figure 2. Table 2 presents the dataset statistics for the benchmark dataset of MultiATIS++ as well as for the newly constructed dataset for disaster NLU.

New Dataset for Disaster NLU. We construct a new intent and slot filling dataset of tweets collected during natural disasters, in two languages: English (en) and Haitian Creole (ht). The tweets originally were released by Appen.[4] For English, a language expert labeled the tweets, and for Haitian Creole, we used Amazon Mechanical Turk with five annotators. Intent classes include: '*request*' and '*others*'. Slot filling consists of 5 labels: '*medical_help*', '*food*', '*water*', '*shelter*', and '*other_aid*'. Table 2 provides the dataset statistics.

4 Experimental Setup

We use the traditional cross-lingual task setting where each experiment consists of a source language and a target language. A model is trained on the source data (English) and evaluated on the target data (8 other languages). For code-switching experiments, the English dataset is augmented with multilingual code-switching before training. Our

[2]Each of the Sino-Tibetan, Koreanic, and Japonic families have a single high-resource member (Chinese, Korean, Japanese respectively). We only group them as an additional interesting data point, not because we ascribe to any theories that link them typologically.

[3]$L = -\frac{1}{n}\sum_{i=1}^{n}[y \log \hat{y}]$

[4]https://appen.com/datasets/combined-disaster-response-data/

Language	Utterances			Tokens			Intents	Slots
	train	dev	test	train	dev	test		
MultiATIS++ (Xu et al., 2020)								
English	4488	490	893	50755	5445	9164	18	84
Spanish	4488	490	893	55197	5927	10338	18	84
Portuguese	4488	490	893	55052	5909	10228	18	84
German	4488	490	893	51111	5517	9383	18	84
French	4488	490	893	55909	5769	10511	18	84
Chinese	4488	490	893	88194	9652	16710	18	84
Japanese	4488	490	893	133890	14416	25939	18	84
Hindi	1440	160	893	16422	1753	9755	17	75
Turkish	578	60	715	6132	686	7683	17	71
Disaster Tweets (New Dataset)								
English	3518	490	-	16369	4242	-	2	5
Haitian Creole	-	-	520	-	-	2834	2	5

Table 2: Datasets & Statistics.

implementation is in PyTorch (Paszke et al., 2019) and we use the pre-trained *bert-base-multilingual-uncased* with *BertForSequenceClassification* (Wolf et al., 2020) model. Maximum epochs is set to 25 with an early stopping patience of 5, batch size of 32, and Adam optimizer (Kingma and Ba, 2014) with a learning rate of 5e$-$5. We select the best model on the validation set. Consistent with the metrics reported for intent prediction and slot filling evaluation in the past, we also accuracy for intent and micro F1[5] to measure slot performance.

4.1 Baselines & Upper Bound

Since we assume that target language is not known before hand, **Translate-Train (TT)** (Xu et al., 2020) method is not a suitable baseline. Rather, we set this to be an upper bound, i.e. translating to the target language and fine-tuning the model should intuitively outperform a generic model. Additionally, we add code-switching to this TT model to assess if augmentation negatively impacts its performance. The zero-shot baselines for the code-switching experiments use an **English-Only** (Xu et al., 2020) model, which is fine-tuned over the pre-trained mBERT separately for each task and an English-only **Joint** model (Chen et al., 2019).

5 Results & Discussion

Effect of Multilingual Code-Switching. Table 3 describes performance evaluation on the MultiATIS++ dataset. When compared to the state-of-the-art jointly trained English-only baseline, we see a $+4.2\%$ boost in intent accuracy and $+1.8\%$ boost in slot F1 scores on average by augmenting the dataset via multilingual code-switching **without requiring the target language**. From the significance tests, except for Spanish and German, all other languages were helped by code-switching for intent detection. For slot filling, improvement on Portuguese and French is insignificant. This suggests that code-switching primarily helped languages that are morphologically more different from the source language (English). For example, Hindi and Turkish have the highest intent performance improvement of $+16.1\%$ and $+9.8\%$ respectively. And for slots, Hindi and Chinese with $+6.0\%$ and $+4.3\%$ respectively. Japanese showed $+4\%$ improvement for intent and $+3.4\%$ for slots.

The runtime of the models in Table 5 (Appendix B) shows that code-switching is expensive, taking up to five hours for five augmentation rounds ($k = 5$). This is because there are k times more data compared to the monolingual source data. Increasing the number of code-switchings (k) for a sentence from 5 to 50 improves the performance by $+1\%$, while increasing the run-time by a large margin. Hence, such tradeoffs should be considered when picking k for real-world applications where time to deployment might be of the essense.

In the translate-train (upper bound) scenario, it is not immediately clear if augmentation helps, since data in the same language as the target are always preferable to other language or code-switched

[5]To address class imbalance for slots, we use Micro F1 instead of Macro F1, which is why our F1 scores are inflated when compared to scores in (Xu et al., 2020).

Intent Acc.	m	es	de	zh	ja	pt	fr	hi	tr	**AVG**
English-Only Baseline*	1	94.42	94.29	79.53	73.75	92.90	93.86	67.06	69.71	83.19
$Joint_{en-only}$ Baseline*	1	95.03	94.51	80.54	73.57	93.48	93.33	73.53	71.05	84.38
Word-level CS[†]	1	94.18	93.92	81.67	75.48	92.54	94.18	81.19	74.22	85.92
Sentence-level CS	1	94.60	93.53	81.21	75.01	93.10	93.24	82.37	75.11	86.02
Chunk-level CS (CCS)	1	95.12	**95.27**	83.88	74.27	**94.20**	93.48	82.73	77.51	87.06
$Joint_{en-only}$* + CCS	1	**95.48**	94.51	**84.43**$^{\spadesuit}$	**76.48**$^{\spadesuit}$	94.15$^{\spadesuit}$	**94.89**$^{\spadesuit}$	**85.37**$^{\spadesuit}$	**78.04**$^{\spadesuit}$	**87.92**
Upper Bound										
Translate-Train (TT)*	8	94.02	93.84	90.21	84.19	95.66	94.54	85.08	85.79	90.42
$Joint_{TT}$*	8	94.16	94.24	91.56	85.98	95.75	95.01	86.45	84.95	91.01
$Joint_{TT}$* + CCS	8	95.48	95.41	91.60	87.17	95.34	94.60	87.94	85.93	91.68
Slot F1	m	es	de	zh	ja	pt	fr	hi	tr	**AVG**
English-Only Baseline*	1	96.16	96.73	83.12	78.81	95.63	95.40	77.05	88.09	88.87
$Joint_{en-only}$ Baseline*	1	96.12	96.76	84.95	79.60	95.76	95.76	77.63	88.92	89.44
Word-level CS[†]	1	95.81	96.33	85.46	79.33	96.27	95.08	79.10	86.86	89.28
Sentence-level CS	1	96.57	**96.92**	86.32	79.52	**96.65**	95.84	81.94	89.84	90.45
Chunk-level CS (CCS)	1	**96.68**	96.82	87.10	80.00	96.46	**96.31**	80.95	**91.60**	90.51
$Joint_{en-only}$* + CCS	1	96.09	96.56	**88.61**$^{\spadesuit}$	**82.28**$^{\spadesuit}$	96.01	95.94	**82.28**$^{\spadesuit}$	90.45$^{\spadesuit}$	**91.03**
Upper Bound										
Translate-Train (TT)*	8	96.89	96.04	93.48	85.29	96.35	96.02	82.03	91.21	92.16
$Joint_{TT}$*	8	96.92	95.66	93.64	87.84	96.11	95.95	82.98	91.15	92.53
$Joint_{TT}$* + CCS	8	96.98	96.27	93.37	85.87	95.88	95.44	82.00	91.31	92.14

Table 3: Performance evaluation of code-switching with setting $k = 5$. CS: Code-Switching. Reported scores are average of 5 independent runs (including a separate code-switched data for each run). m = number of distinct models to be trained. *: modified BERT-based implementations (Chen et al., 2019; Xu et al., 2020). †: Similar to Qin et al., 2020 but modified for slot-filling task and also excluding target language from randomized switching. $^{\spadesuit}$: The difference is significant with $p < 0.05$ using Tukey HSD (conducted between $Joint_{en-only}$ + CCS versus $Joint_{en-only}$ Baseline for each language).

Intent Acc.	ht
English-Only Baseline*	56.12
Translate-Train (TT)	62.58
Chunk-level CS (CCS)	63.15
$Joint_{en-only}$* + CCS	**63.73**
Slot F1	ht
English-Only Baseline*	68.72
Translate-Train (TT)	69.96
Chunk-level CS (CCS)	**70.27**
$Joint_{en-only}$* + CCS	70.02

Table 4: Performance on disaster data in Haitian Creole (ht). CS = Code-Switching. Reported scores are average of 5 independent runs (*: modified BERT-based).

data. At a minimum, augmentation does not hinder upper-bound performance (Table 3).

For both intent and slot performance, the chunk-level model remained robust across the languages. For intent, the difference between word-level and sentence-level was insignificant. For slot, sentence-level was in par with chunk-level on average. Thus, we think that code-switching at chunk-level is safer for avoiding semantic discrepancies (which are a danger in the word-level) while also better encouraging intra-sentence language neutrality.

Evaluation on Disaster Dataset. We found that disaster data is more challenging than the ATIS dataset for transfer learning in NLU. The predictive performance is shown in Table 4. Code-Switching improved intent accuracy by $+12.5\%$ and slot F1 by $+2.3\%$, which is quite promising considering the domain mismatch (tweets vs airline guides). Joint training added $+0.9\%$ improvement to intent accuracy, however did not seem to help slot F1. This might imply a weaker correlation between the two tasks in real-world data, i.e. a mention of '*food*' or '*shelter*' in a tweet may not always mean that there is a '*request*' or vice-versa. The upper bound of translate-train method did not perform any better than the randomly code-switched model which seemed counter-intuitive. This might be due to the lack of strong representation for Haitian Creole in the pre-trained model, although it is similar to French, or due to the limitation of the machine translation system.

Impact of Language Families. Results of language family analysis are shown in Figure 3 for the 4 languages that showed significant improvements for both intent and slots in Table 3. The input in English is independently code-switched using 6 different language families. Note that the target language is always excluded from the group when evaluating on the same, i.e. Hindi is excluded from Indo-Aryan family when that family is being evaluated on it. Translate-train model is provided as a frame of reference and upper bound. Generally, as expected, we found that language families helped their corresponding languages, i.e. Romance helped Spanish, Germanic helped German,

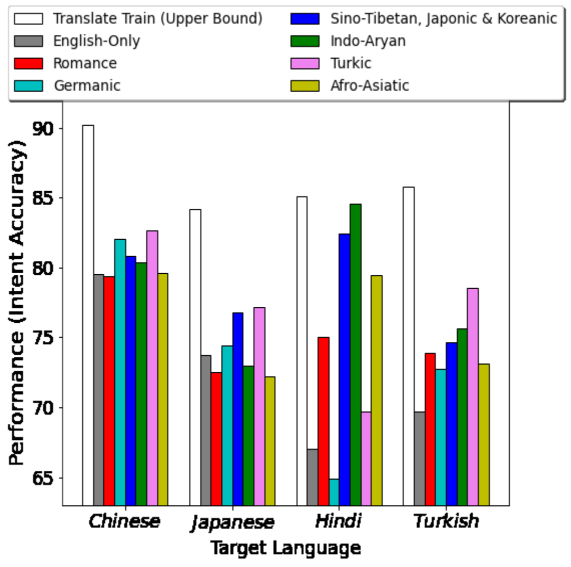


Figure 3: Impact of different language groups on the target languages.

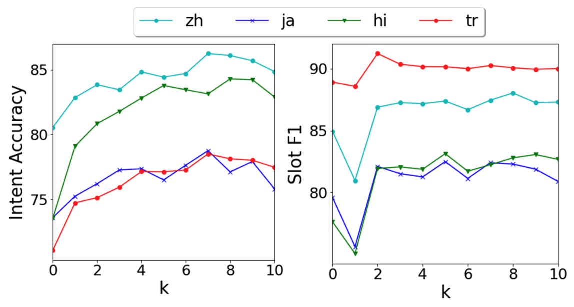


Figure 4: Performance as k (augmentation rounds) increases (on mBERT).

and so on. An exception is our loose group of Sino-Tibetan, Koreanic, and Japonic languages –for both Chinese and Japanese, languages from the Turkic language family helped more than others. On the other hand, the Sino-tibetan, Japonic, and Koreanic group helps Hindi more than other Indo-European languages. We believe this highlights the necessity for methods like the one of Xia et al. (2020) that can *a priori* identify the best helper language or group of languages that can benefit downstream tasks for low resource languages.

Control Experiments on k. Hyperparameter k controls the amount of code-switched data. $k = 0$ represents original size with no code-switching, $k = 1$ represents original size with code-switching, and $k = 10$ means 10-times more code-switched data than the original. The main experiments in Table 3 use $k = 5$. Figure 4 shows how varying k affects performance. For this analysis, we consider 4 target languages on which code-switching produced significant results in Table 3 on both Intent Accuracy and Slot F1: Chinese, Japanese, Hindi, and Turkish. Intuitively, we observe that as k increases, too much code-switching becomes expensive in terms of runtime, while performance improvement slowly plateaus. For Slot F1 performance in all four cases, unlike Intent, we observe an interesting dip when $k = 1$, which represents the augmentation having just one copy of code-switching (without the original non-code-switched data), as compared to $k = 0$. Adding the original data to one round of code-switched data ($k = 2$) leads to big improvements. Overall, we see improvement for both tasks, with Slot F1 plateauing earlier. Table 5 and Figure 10 in Appendix B show the impact of code-switching on training runtime, which increases as k increases. Thus, finding an optimal value of k and specific language groups are essential for downstream applications.

mBERT versus XLM-R. Additional performance evaluations and benefits of code-switching on XLM-R (Conneau et al., 2020a), a stronger multilingual language model, are provided in Appendix A. Note that XLM-R is trained using Common-Crawl and is likely to be exposed to some code-switched data. Thus, we focus primarily on mBERT which largely remains monolingual at the sentence-level to identify the unbiased impact of code-switching during *fine-tuning*. Furthermore, runtime and hyperparameter tuning along with insights into layers to freeze before training are shown in Appendix B.

Error Analysis. Selecting intent classes with support > 10, Figure 5 shows how each class is positively or negatively impacted by code-switching. Improvement was primarily on '*airfare*', '*distance*' '*capacity*', '*airline*', and '*ground_service*' which had longer sentences such as '*Please tell me which airline has the most departures from Atlanta*' when compared to '*abbreviations*' and '*airport*' classes that included very short phrases like '*What does EA mean?*' However, note that Spanish and German did not improve much, aligning with our results in Table 3. For slot labels in Figure 6, we selected the ones with support > 50 and that have different characteristics, e.g. '*name*', '*code*', etc. The overall trend in slot performance shows improvements for labels such as '*day_name*', '*airport_code*', and '*city_name*' and slight variations in labels such as '*fight_number*' and '*period_of_day*', implying textual slots benefiting over numeric ones.

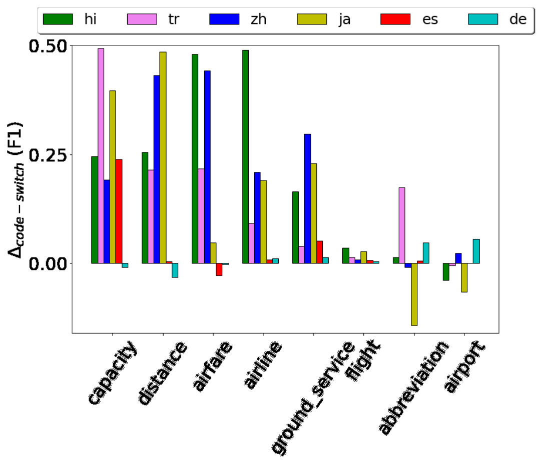
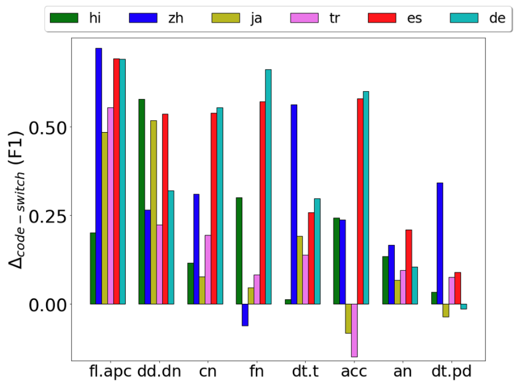


Figure 5: Impact of code-switching on intent classes.



acc: B-aircraft_code, ***dt.t***: B-depart_time.time, ***cn***: B-city_name, ***an***: B-airline_name, ***dt.pd***: B-depart _time.period_of_day, ***dd.dn***: B-depart_date .day_name, ***fn***: B-flight_number, ***fl.apc***: B-fromloc.airport_code

Figure 6: Impact of code-switching on slot labels.

6 Related Work

Cross-Lingual Transfer. Researchers have studied cross-lingual tasks in various settings such as sentiment/sequence classification (Wan, 2009; Eriguchi et al., 2018; Yu et al., 2018), named entity recognition (Zirikly and Hagiwara, 2015; Tsai et al., 2016; Xie et al., 2018), parts-of-speech tagging (Yarowsky et al., 2001; Täckström et al., 2013; Plank and Agić, 2018), and natural language understanding (He et al., 2013; Upadhyay et al., 2018; Xu et al., 2020). The methodology for most of the current approaches for cross-lingual tasks can be categorizes as: **a)** multilingual representations from pre-trained or fine-tuned models such as mBERT (Devlin et al., 2019) or XLM-R (Conneau et al., 2020a), **b)** machine translation followed by alignment (Shah et al., 2010; Yarowsky et al., 2001; Ni et al., 2017), or **c)** a combination of both (Xu et al., 2020). Before transformer models, effective approaches included domain adversarial training to extract language-agnostic features (Ganin et al., 2016; Chen et al., 2018) and word alignment methods such as MUSE (Conneau et al., 2017) to align fastText word vectors (Bojanowski et al., 2017). Recently, Conneau et al., 2020b show that having shared parameters in the top layers of the multilingual encoders can be used to align different languages quite effectively on tasks such as XNLI (Conneau et al., 2018).

Monolingual models for joint slot filling and intent prediction have used attention-based RNN (Liu and Lane, 2016) and attention-based BiLSTM with a slot gate (Goo et al., 2018) on benchmark datasets (Price, 1990; Coucke et al., 2018). These methods have shown that a joint method can enhance both tasks and slot filling can be conditioned on the learned intent. A related approach iteratively learns the relationship between the two tasks (Haihong et al., 2019) . Recently, BERT-based approaches (Hardalov et al., 2020; Chen et al., 2019) have improved results. On the other hand, cross-lingual versions of this joint task include a low-supervision based approach for Hindi and Turkish (Upadhyay et al., 2018), new datasets for Spanish and Thai (Schuster et al., 2019), and recently Xu et al. (2020) creating MultiATIS++, a comprehensive dataset in 9 languages. The joint task mentioned above in a pure zero-shot setting is one of the motivations for our work. A Zero-shot is the setting where the model sees a new distribution of examples only during test (prediction) time (Xian et al., 2017; Srivastava et al., 2018; Romera-Paredes and Torr, 2015). Thus, in our setting, we assume that target language is unknown during training, so that our model is generalizable across multiple languages.

Code-Switching. Linguistic code-switching is a phenomenon where multilingual speakers alternate between languages. Recently, monolingual models have been adapted to code-switched text in entity recognition (Aguilar and Solorio, 2019), part-of-speech tagging (Soto and Hirschberg, 2018; Ball and Garrette, 2018), sentiment analysis (Joshi et al., 2016) and language identification (Mave et al., 2018; Yirmibeşoğlu and Eryiğit, 2018; Mager et al., 2019). Recently, KhudaBukhsh et al., 2020 have proposed an approach to sample code-mixed documents using minimal supervision. Qin et al., 2020 allows randomized code-switching to include the target language, as shown in their Figure 3. In our context for example, if the target language is German, we ensure that there is no code-switching to German during training. We consider this distinction essential to evaluate a true zero-shot learning scenario and prevent any bias when comparing with translate-and-train. Yang et al. (2020) present a non-zero-shot approach that performs code-switching to target languages, and Jiang et al. (2020) present a code-switching based method to improve the ability of multilingual language models for factual knowledge retrieval. Contemporary work by Tan and Joty, 2021 makes use of both word and phrase-level code-mixing to switch to a set of languages to perform adversarial training for XNLI. Code-switching and other data augmentation techniques have been applied to the pre-training stage in recent works (Chaudhary et al., 2020; Kale and Siddhant, 2021; Dufter and Schütze, 2020). However, *pre-training* is outside the scope of this work. In addition to studying cross-lingual slot filling and language families, another key distinction of our method is that we completely ignore the target language during training to represent a fully zero-shot scenario. The main advantage is that with enhanced cross-lingual generalizability, it can be deployed out-of-the-box, as our training is conducted independently of the target language.

7 Conclusion & Future Work

Our study shows that augmenting the monolingual input data with multilingual code-switching via random translations at the chunk-level helps a zero-shot model to be language neutral when evaluated on unseen languages. This approach enhanced the generalizability of pre-trained language models such as mBERT when *fine-tuning* for downstream tasks of intent detection and slot filling. Additionally, we presented an application of this method using a new annotated dataset of disaster tweets. Further, we studied code-switching with language families and their impact on specific target languages. Addressing code-switching with language families during the pre-training phase and releasing a larger dataset of annotated disaster tweets in more languages are planned for future work.

8 Ethical Considerations

The tweet dataset that we constructed for disaster NLU was originally released by Appen[6], and we use it to construct slot labels in two languages: English (en) and Haitian Creole (ht). Data statement that includes annotator guidelines for the labeling jobs and other dataset information will be provided with the implementation. From a broader impact perspective, our code and developed models are open-source and allows NLP technology to be accessible to information systems for emergency services and social scientists in quickly deploying model during disaster events.

[6]https://appen.com/datasets/combined-disaster-response-data/

Acknowledgements

This work was partially supported by U.S. National Science Foundation grants IIS-1815459, IIS-1657379, and 2040926. This work was also supported in part by the grant H-4Q21-009 from the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development (for more information about CCI, visit www.cyberinitiative.org). The authors are thankful to Ming Sun and Alexis Conneau for giving valuable insights on multilingual model training, as well as to the anonymous reviewers for their constructive feedback. We also acknowledge ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, were most experiments were conducted.

References

Gustavo Aguilar and Thamar Solorio. 2019. From english to code-switching: Transfer learning with strong morphological clues. *arXiv preprint arXiv:1909.05158*.

Kelsey Ball and Dan Garrette. 2018. Part-of-speech tagging for code-switched, transliterated texts without explicit language identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3084–3089.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Aditi Chaudhary, Karthik Raman, Krishna Srinivasan, and Jiecao Chen. 2020. Dict-mlm: Improved multilingual pre-training using bilingual dictionaries. *arXiv preprint arXiv:2010.12566*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034, Online. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

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.

Philipp Dufter and Hinrich Schütze. 2020. Identifying elements essential for bert's multilinguality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4423–4437.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

William Denis Elcock and John N Green. 1960. *The romance languages*. Faber & Faber London.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *arXiv preprint arXiv:1809.04686*.

Jack GM FitzGerald. 2020. Stil–simultaneous slot filling, translation, intent classification, and language identification: Initial results using mbart on multiatis++. *arXiv preprint arXiv:2010.00760*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

E Haihong, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5467–5471.

Wayne Harbert. 2006. *The Germanic Languages*. Cambridge University Press.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2020. Enriched pre-trained transformers for joint slot filling and intent detection. *arXiv preprint arXiv:2004.14848*.

Xiaodong He, Li Deng, Dilek Hakkani-Tur, and Gokhan Tur. 2013. Multi-style adaptive training for robust cross-lingual spoken language understanding. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8342–8346. IEEE.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020. Multilingual factual knowledge retrieval from pretrained language models. *arXiv preprint arXiv:2010.06189*.

Lars Johanson and Éva Ágnes Csató Johanson. 2015. *The Turkic Languages*. Routledge.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

Mihir Kale and Aditya Siddhant. 2021. Mixout: A simple yet effective data augmentation scheme for slot-filling. In *Conversational Dialogue Systems for the Next Decade*, pages 279–288. Springer.

Ashiqur R KhudaBukhsh, Shriphani Palakodety, and Jaime G Carbonell. 2020. Harnessing code switching to transcend the linguistic barrier. *arXiv preprint arXiv:2001.11258*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jitin Krishnan, Hemant Purohit, and Huzefa Rangwala. 2020. Unsupervised and interpretable domain adaptation to rapidly filter social web data for emergency services. In *ASONAM*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Manuel Mager, Özlem Çetinoğlu, and Katharina Kann. 2019. Subword-level language identification for intra-word code-switching. 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 2005–2011.

Colin P Masica. 1993. *The indo-aryan languages*. Cambridge University Press.

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.

Roy Andrew Miller. 1967. *The Japanese Language*. University of Chicago Press Chicago.

Aldrian Obaja Muis, Naoki Otani, Nidhi Vyas, Ruochen Xu, Yiming Yang, Teruko Mitamura, and Eduard Hovy. 2018. Low-resource cross-lingual event type detection via distant supervision with minimal effort. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 70–82.

Dat Tien Nguyen, Kamla Al-Mannai, Shafiq R Joty, Hassan Sajjad, Muhammad Imran, and Prasenjit Mitra. 2017. Robust classification of crisis-related data on social networks using convolutional neural networks. *ICWSM*, 31(3):632–635.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1470–1480.

Endang Wahyu Pamungkas and Viviana Patti. 2019. Cross-domain and cross-lingual abusive language detection: A hybrid approach with deep learning and a multilingual lexicon. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 363–370.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001.

Barbara Plank and Željko Agić. 2018. Distant supervision from disparate sources for low-resource part-of-speech tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 614–620.

Patti Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Libo Qin, Minheng Ni, Yue Zhang, and Wanxiang Che. 2020. Cosda-ml: Multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp. *arXiv preprint arXiv:2006.06402*.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161.

Bruce Rowe and Diane Levine. 2017. A concise introduction to linguistics. *Routledge. pp. 340–341*.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2019. Cross-lingual transfer learning for multilingual task oriented dialog. 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 3795–3805.

Robert Shafer. 1955. Classification of the sino-tibetan languages. *Word*, 11(1):94–111.

Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2010. Synergy: a named entity recognition system for resource-scarce languages such as swahili using online machine translation. In *Proceedings of the Second Workshop on African Language Technology (AfLaT 2010)*, pages 21–26.

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.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316.

Lukas Stappen, Fabian Brunn, and Björn Schuller. 2020. Cross-lingual zero-and few-shot hate speech detection utilising frozen transformer language models and axel. *arXiv preprint arXiv:2004.13850*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Samson Tan and Shafiq Joty. 2021. Code-mixing on sesame street: Dawn of the adversarial polyglots. *arXiv preprint arXiv:2103.09593*.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 219–228.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Hakkani-Tür Dilek, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038. IEEE.

Charles Frederick Voegelin and Florence Marie Voegelin. 1976. Classification and index of the world's languages.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 235–243.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Mengzhou Xia, Antonios Anastasopoulos, Ruochen Xu, Yiming Yang, and Graham Neubig. 2020. Predicting performance for natural language processing tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8625–8646, Online. Association for Computational Linguistics.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime G Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379.

Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-end slot alignment and recognition for cross-lingual nlu. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5052–5063.

Jian Yang, Shuming Ma, Dongdong Zhang, ShuangZhi Wu, Zhoujun Li, and Ming Zhou. 2020. Alternating language modeling for cross-lingual pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9386–9393.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. Technical report, Johns Hopkins Univ Baltimore MD Dept of Computer Science.

Zeynep Yirmibeşoğlu and Gülşen Eryiğit. 2018. Detecting code-switching between turkish-english language pair. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 110–115.

Katherine Yu, Haoran Li, and Barlas Oguz. 2018. Multilingual seq2seq training with similarity loss for cross-lingual document classification. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 175–179.

Ayah Zirikly and Masato Hagiwara. 2015. Cross-lingual transfer of named entity recognizers without parallel corpora. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 390–396.

A mBERT versus XLM-R

We conduct an additional analysis on XLM-R (Conneau et al., 2020a) and compare it with mBERT (Devlin et al., 2019). The implementation is very similar in PyTorch (Paszke et al., 2019) but using the pre-trained *xlm-roberta-base* with *RobertaForSequenceClassification* (Wolf et al., 2020) as the XLM-R model. We observe that, setting $k = 5$, XLM-R outperforms mBERT on average (by 2% Intent Accuracy and 1.5% Slot F1). Individually, XLM-R improved Chinese, Japanese, Portuguese, and Turkish for Intent Prediction and German, Chinese, Japanese, Portuguese, and Hindi for Slot Filling as shown in Figure 7. We observe a trend similar to mBERT with k on XLM-R shown in Figure 8. However, for XLM-R, we observe that randomized code-switching did not help Chinese for Intent Prediction and Hindi for Slot F1. If code-switched to a specific language family, instead of switching to random languages, it might improve their performance. A deeper dive into XLM-R and language families are left for future work.

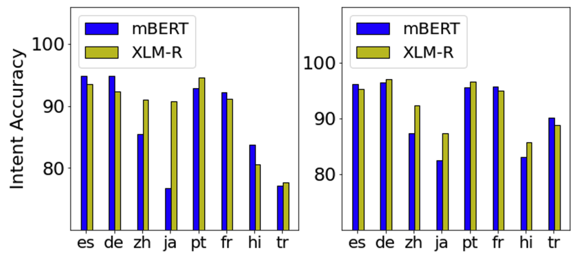


Figure 7: mBERT vs XLM-R: Performance

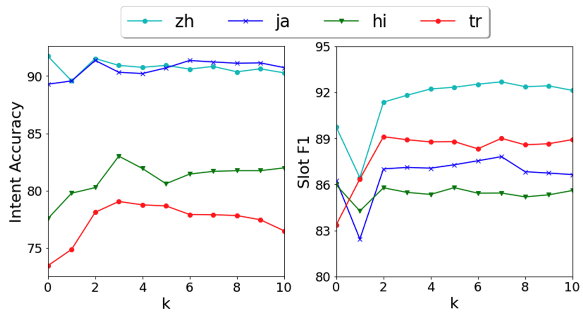


Figure 8: Performance as k increases (on XLM-R).

B Hyperparameter Tuning & Runtime

For joint training with same task weights, we tuned α and β using grid search to see the strength of correlation between the tasks. For intent, the (α, β) combination of $(1.0, 0.6)$ performed well, while $(1.0, 1.0)$ for slots. This suggests that intent benefiting slot might be slightly more than slot benefiting intent. Additionally, during fine-tuning, freezing the layers of the transformer affected the model performance as shown in Figure 9. Keeping the first 8 layers frozen gave the best performance. By freezing the earlier layers, the transformer can retain its most fundamental feature information gained from the massive pre-training step, and by unfreezing some top layers, it can undergo fine-tuning. Additionally, latency for training a code-switched model is shown in Table 5 and how runtime varies with an increase in k is shown in Figure 10.

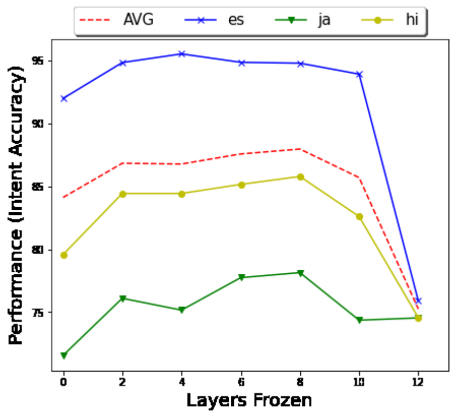


Figure 9: Freezing earlier layers and unfreezing a few at the top of the transformer appear to be most optimal.

CS (k=5)	MTT	Joint_{en}	Joint_{cs}	Joint_{TT}
05:04:49	1:31:32	00:11:50	01:06:50	00:11:04

Table 5: Runtime on Google Colab (K80 GPU for training joint models). MTT: Machine Translation to Target. Note that MTT and J_{TT} are for one target language (averaged).

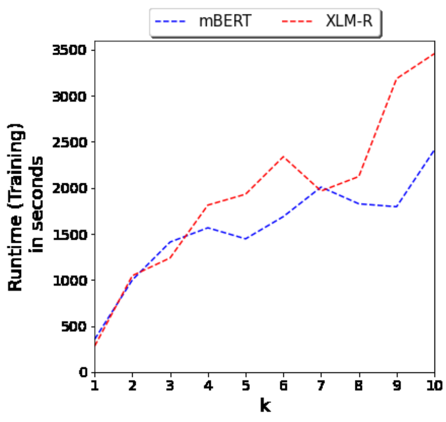


Figure 10: Training runtime as k increases.