Are Multilingual Models Effective in Code-Switching?

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin,
Andrea Madotto, Pascale Fung
Center for Artificial Intelligence Research (CAiRE)
The Hong Kong University of Science and Technology
giwinata@connect.ust.hk

Abstract

Multilingual language models have shown decent performance in multilingual and cross-lingual natural language understanding tasks. However, the power of these multilingual models in code-switching tasks has not been fully explored. In this paper, we study the effectiveness of multilingual language models to understand their capability and adaptability to the mixed-language setting by considering the inference speed, performance, and number of parameters to measure their practicality. We conduct experiments in three language pairs on named entity recognition and part-of-speech tagging and compare them with existing methods, such as using bilingual embeddings and multilingual meta-embeddings. Our findings suggest that pre-trained multilingual models do not necessarily guarantee high-quality representations on code-switching, while using meta-embeddings achieves similar results with significantly fewer parameters.

1 Introduction

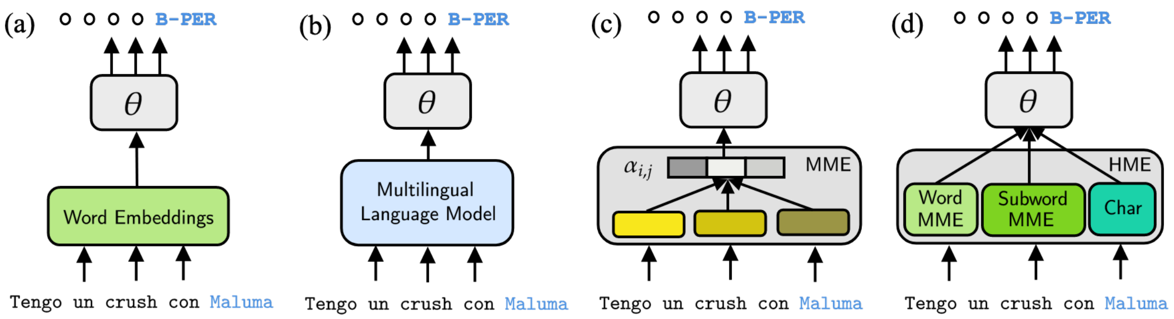
Learning representation for code-switching has become a crucial area of research to support a greater variety of language speakers in natural language processing (NLP) applications, such as dialogue system and natural language understanding (NLU). Code-switching is a phenomenon in which a person speaks more than one language in a conversation, and its usage is prevalent in multilingual communities. Yet, despite the enormous number of studies in multilingual NLP, only very few focus on code-switching. Recently, contextualized language models, such as mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) have achieved state-of-the-art results on monolingual and cross-lingual tasks in NLU benchmarks (Wang et al., 2018a; Hu et al., 2020; Wilie et al., 2020; Liu et al., 2020; Lin et al., 2020). However, the effectiveness of these multilingual language models on code-switching tasks remains unknown.

Several approaches have been explored in code-switching representation learning in NLU. Character-level representations have been utilized to address the out-of-vocabulary issue in code-switched text (Winata et al., 2018c; Wang et al., 2018b), while external handcrafted resources such as gazetteers list are usually used to mitigate the low-resource issue in code-switching (Aguilar et al., 2017; Trivedi et al., 2018); however, this approach is very limited because it relies on the size of the dictionary and it is language-dependent. In another line of research, meta-embeddings have been used in code-switching by combining multiple word embeddings from different languages (Winata et al., 2019a,b). This method shows the effectiveness of mixing word representations in closely related languages to form language-agnostic representations, and is considered very effective in Spanish-English code-switched named entity recognition tasks, and significantly outperforming mBERT (Khanuja et al., 2020) with fewer parameters.

While more advanced multilingual language models (Conneau et al., 2020) than multilingual BERT (Devlin et al., 2019) have been proposed, their effectiveness is still unknown in code-switching tasks. Thus, we investigate their effectiveness in the code-switching domain and compare them with the existing works. Here, we would like to answer the following research question, *"Which models are effective in representing code-switching text, and why?."*

In this paper, we evaluate the representation quality of monolingual and bilingual word embeddings, multilingual meta-embeddings, and multilingual language models on five downstream tasks on named entity recognition (NER) and part-of-speech tagging (POS) in Hindi-English, Spanish-English, and Modern Standard Arabic-Egyptian. We study the effectiveness of each model by considering three criteria: performance, speed, and the

Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching, pages 142–153
June 11, 2021. ©2021 Association for Computational Linguistics
https://doi.org/10.26615/978-954-452-056-4_020



(a)
O O O O B-PER
θ
Word Embeddings
Tengo un crush con Maluma
(b)
O O O O B-PER
θ
Multilingual Language Model
Tengo un crush con Maluma
(c)
O O O O B-PER
θ
$\alpha_{i,j}$
MME
Tengo un crush con Maluma
(d)
O O O O B-PER
θ
HME
Word MME
Subword MME
Char
Tengo un crush con Maluma

Figure 1: Model architectures for code-switching modeling: (a) model using word embeddings, (b) model using multilingual language model, (c) model using multilingual meta-embeddings (MME), and (d) model using hierarchical meta-embeddings (HME).

number of parameters that are essential for practical applications. Here, we set up the experimental setting to be as language-agnostic as possible; thus, it does not include any handcrafted features.

Our findings suggest that multilingual pre-trained language models, such as XLM-R$_{\text{BASE}}$, achieves similar or sometimes better results than the hierarchical meta-embeddings (HME) (Winata et al., 2019b) model on code-switching. On the other hand, the meta-embeddings use word and subword pre-trained embeddings that are trained using significantly less data than mBERT and XLM-R$_{\text{BASE}}$ and can achieve on par performance to theirs. Thus, we conjecture that the masked language model is not be the best training objective for representing code-switching text. Interestingly, we found that XLM-R$_{\text{LARGE}}$ can improve the performance by a great margin, but with a substantial cost in the training and inference time, with 13x more parameters than HME-Ensemble for only around a 2% improvement. The main contributions of our work are as follows:

- We evaluate the performance of word embeddings, multilingual language models, and multilingual meta-embeddings on code-switched NLU tasks in three language pairs, Hindi-English (HIN-ENG), Spanish-English (SPA-ENG), and Modern Standard Arabic-Egyptian (MSA-EA), to measure their ability in representing code-switching text.
- We present a comprehensive study on the effectiveness of multilingual models on a variety of code-switched NLU tasks to analyze the practicality of each model in terms of performance, speed, and number of parameters.
- We further analyze the memory footprint required by each model over different sequence lengths in a GPU. Thus, we are able to understand which model to choose in a practical scenario.

2 Representation Models

In this section, we describe multilingual models that we explore in the context of code-switching. Figure 1 shows the architectures for a word embeddings model, a multilingual language model, and the multilingual meta-embeddings (MME), and HME models.

2.1 Word Embeddings

2.1.1 FastText

In general, code-switching text contains a primary language the matrix language (ML)) as well as a secondary language (the embedded language (EL)). To represent code-switching text, a straightforward idea is to train the model with the word embeddings of the ML and EL from FastText (Grave et al., 2018). Code-switching text has many noisy tokens and sometimes mixed words in the ML and EL that produce a "new word", which leads to a high number of out-of-vocabulary (OOV) tokens. To solve this issue, we utilize subword-level embeddings from FastText (Grave et al., 2018) to generate the representations for these OOV tokens. We conduct experiments on two variants of applying the word embeddings to the code-switching tasks: FastText (ML) and FastText (EL), which utilize the word embeddings of ML and EL, respectively.

2.1.2 MUSE

To leverage the information from the embeddings of both the ML and EL, we utilize MUSE (Lample et al., 2018) to align the embeddings space of the ML and EL so that we can inject the information of the EL embeddings into the ML embeddings, and vice versa. We perform alignment in two directions: (1) We align the ML embeddings to the vector space of the EL embeddings (denoted as MUSE (ML $\rightarrow$ EL)); (2) We conduct the alignment in the opposite direction, which aligns the EL embeddings to the vector space of the ML embeddings (denoted as MUSE (EL $\rightarrow$ ML)). After the embeddings alignment, we train the model with the aligned embeddings for the code-switching tasks.

2.2 Multilingual Pre-trained Models

Pre-trained on large-scale corpora across numerous languages, multilingual language models (Devlin et al., 2019; Conneau et al., 2020) possess the ability to produce aligned multilingual representations for semantically similar words and sentences, which brings them advantages to cope with code-mixed multilingual text.

2.2.1 Multilingual BERT

Multilingual BERT (mBERT) (Devlin et al., 2019), a multilingual version of the BERT model, is pre-trained on Wikipedia text across 104 languages with a model size of 110M parameters. It has been shown to possess a surprising multilingual ability and to outperform existing strong models on multiple zero-shot cross-lingual tasks (Pires et al., 2019; Wu and Dredze, 2019). Given its strengths in handling multilingual text, we leverage it for code-switching tasks.

2.2.2 XLM-RoBERTa

XLM-RoBERTa (XLM-R) (Conneau et al., 2020) is a multilingual language model that is pre-trained on 100 languages using more than two terabytes of filtered CommonCrawl data. Thanks to the large-scale training corpora and enormous model size (XLM-R$_{\text{BASE}}$ and XLM-R$_{\text{LARGE}}$ have 270M and 550M parameters, respectively), XLM-R is shown to have a better multilingual ability than mBERT, and it can significantly outperform mBERT on a variety of cross-lingual benchmarks. Therefore, we also investigate the effectiveness of XLM-R for code-switching tasks.

2.2.3 Char2Subword

Char2Subword introduces a character-to-subword module to handle rare and unseen spellings by training an embedding lookup table (Aguilar et al., 2020b). This approach leverages transfer learning from an existing pre-trained language model, such as mBERT, and resumes the pre-training of the upper layers of the model. The method aims to increase the robustness of the model to various typography styles.

2.3 Multilingual Meta-Embeddings

The MME model (Winata et al., 2019a) is formed by combining multiple word embeddings from different languages. Let's define $\mathbf{w}$ to be a sequence of words with n elements, where $\mathbf{w} = [w_1, \ldots, w_n]$. First, a list of word-level embedding layers is used $E_i^{(w)}$ to map words $\mathbf{w}$ into embeddings $\mathbf{x}_i$. Then, the embeddings are combined using one out of the following three methods: concat, linear, and self-attention. We briefly discuss each method below.

Concat This method concatenates word embeddings by merging the dimensions of word representations into higher-dimensional embeddings. This is one of the simplest methods to join all embeddings without losing information, but it requires a larger activation memory than the linear method.

$$\mathbf{x}_i^{\text{CONCAT}} = [\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,n}]. \quad (1)$$

Linear This method sums all word embeddings into single word embeddings with equal weight without considering each embedding's importance. The method may cause a loss of information and may generate noisy representations. Also, though it is very efficient, it requires an additional layer to project all embeddings into a single-dimensional space if one embedding is larger than another.

$$\mathbf{x'}_{i,j} = \mathbf{W}_j \cdot \mathbf{x}_{i,j},$$
$$\mathbf{x}_i^{\text{LINEAR}} = \sum_{j=0}^{n} \mathbf{x'}_{i,j}.$$

Self-Attention This method generates a meta-representation by taking the vector representation from multiple monolingual pre-trained embeddings in different subunits, such as word and subword. It applies a projection matrix $\mathbf{W}_j$ to transform the dimensions from the original space $\mathbf{x}_{i,j} \in \mathbb{R}^d$ to a new shared space $\mathbf{x'}_{i,j} \in \mathbb{R}^{d'}$. Then, it calculates attention weights $\alpha_{i,j} \in \mathbb{R}^{d'}$ with a non-linear scoring function ϕ (e.g., tanh) to take important information from each individual embedding $\mathbf{x'}_{i,j}$. Then, MME is calculated by taking the weighted sum of the projected embeddings $\mathbf{x'}_{i,j}$:

$$\mathbf{x'}_{i,j} = \mathbf{W}_j \cdot \mathbf{x}_{i,j}, \tag{2}$$

$$\alpha_{i,j} = \frac{\exp(\phi(\mathbf{x'}_{i,j}))}{\sum_{k=1}^{n} \exp(\phi(\mathbf{x'}_{i,k}))}, \tag{3}$$

$$\mathbf{u}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{x'}_{i,j}. \tag{4}$$

2.4 Hierarchical Meta-Embedings

The HME method combines word, subword, and character representations to create a mixture of embeddings (Winata et al., 2019b). It generates multilingual meta-embeddings of words and subwords, and then, concatenates them with character-level embeddings to generate final word representations. HME combines the word-level, subword-level, and character-level representations by concatenation, and randomly initializes the character embeddings. During the training, the character embeddings are trainable, while all subword and word embeddings remain fixed.

2.5 HME-Ensemble

The ensemble is a technique to improve the model's robustness from multiple predictions. In this case, we train the HME model multiple times and take the prediction of each model. Then, we compute the final prediction by majority voting to achieve a consensus. This method has shown to be very effective in improving the robustness of an unseen test set (Winata et al., 2019c). Interestingly, this method is very simple to implement and can be easily spawned in multiple machines, as in parallel processes.

3 Experiments

In this section, we describe the details of the datasets we use and how the models are trained.

3.1 Datasets

We evaluate our models on five downstream tasks in the LinCE Benchmark (Aguilar et al., 2020a). We choose three named entity recognition (NER) tasks, Hindi-English (HIN-ENG) (Singh et al., 2018a), Spanish-English (SPA-ENG) (Aguilar et al., 2018) and Modern Standard Arabic (MSA-EA) (Aguilar et al., 2018), and two part-of-speech (POS) tagging tasks, Hindi-English (HIN-ENG) (Singh et al., 2018b) and Spanish-English (SPA-ENG) (Soto and Hirschberg, 2017). We apply Roman-to-Devanagari transliteration on the Hindi-English datasets since the multilingual models are trained with data using that form. Table 1 shows the number of tokens of each language for each dataset. We classify the language with more tokens as the ML and the other as the EL. We replace user hashtags and mentions with `<USR>`, emoji with `<EMOJI>`, and URL with `<URL>` for models that use word-embeddings, similar to Winata et al. (2019a). We evaluate our model with the micro F1 score for NER and accuracy for POS tagging, following Aguilar et al. (2020a).

	#L1	#L2	ML	EL
NER				
HIN-ENG	13,860	11,391	HIN	ENG
SPA-ENG	163,824	402,923	ENG	SPA
MSA-EA[†]	-	-	MSA	EA
POS				
HIN-ENG	12,589	9,882	HIN	ENG
SPA-ENG	178,135	92,517	SPA	ENG

Table 1: Dataset statistics are taken from Aguilar et al. (2020a). We define L1 and L2 as the languages found in the dataset. For example, in HIN-ENG, L1 is HIN and L2 is ENG. [†]We define MSA as ML and EA as EL. #L1 represents the number of tokens in the first language and #L2 represents the number of tokens in the second language.

3.2 Experimental Setup

We describe our experimental details for each model.

3.2.1 Scratch

We train transformer-based models without any pre-training by following the mBERT model structure, and the parameters are randomly initialized, including the subword embeddings. We train transformer models with four and six layers with a hidden size of 768. This setting is important to measure the effectiveness of pre-trained multilingual models. We start the training with a learning rate of 1e-4 and an early stop of 10 epochs.

3.2.2 Word Embeddings

We use FastText embeddings (Grave et al., 2018; Mikolov et al., 2018) to train our transformer models. The model consists of a 4-layer transformer encoder with four heads and a hidden size of 200. We train a transformer followed by a Conditional Random Field (CRF) layer (Lafferty et al., 2001).

The model is trained by starting with a learning rate of 0.1 with a batch size of 32 and an early stop of 10 epochs. We also train our model with only ML and EL embeddings. We freeze all embeddings and only keep the classifier trainable.

We leverage MUSE (Lample et al., 2018) to align the embeddings space between the ML and EL. MUSE mainly consists of two stages: adversarial training and a refinement procedure. For all alignment settings, we conduct the adversarial training using the SGD optimizer with a starting learning rate of 0.1, and then we perform the refinement procedure for five iterations using the Procrustes solution and CSLS (Lample et al., 2018). After the alignment, we train our model with the aligned word embeddings (MUSE (ML → EL) or MUSE (EL → ML)) on the code-switching tasks.

3.2.3 Pre-trained Multilingual Models

We use pre-trained models from Huggingface. [1] On top of each model, we put a fully-connected layer classifier. We train the model with a learning rate between [1e-5, 5e-5] with a decay of 0.1 and a batch size of 8. For large models, such as XLM-R_{LARGE} and XLM-$\text{MLM}_{\text{LARGE}}$, we freeze the embeddings layer to fit in a single GPU.

3.2.4 Multilingual Meta-Embeddings (MME)

We use pre-trained word embeddings to train our MME. Table 2 shows the embeddings used for each dataset. We freeze all embeddings and train a transformer classifier with the CRF. The transformer classifier consists of a hidden size of 200, a head of 4, and 4 layers. All models are trained with a learning rate of 0.1, an early stop of 10 epochs, and a batch size of 32. We follow the implementation from the code repository. [2] Table 2 shows the list of word embeddings used in MME.

3.2.5 Hierarchical Meta-Embeddings (HME)

We train our HME model using the same embeddings as MME and pre-trained subword embeddings from Heinzerling and Strube (2018). The subword embeddings for each language pair are shown in Table 3. We freeze all word embeddings and subword embeddings, and keep the character embeddings trainable.

	Word Embeddings List
NER	
HIN-ENG	FastText: Hindi, English (Grave et al., 2018)
SPA-ENG	FastText: Spanish, English, Catalan, Portugese (Grave et al., 2018) GLoVe: English-Twitter (Pennington et al., 2014)
MSA-EA	FastText: Arabic, Egyptian (Grave et al., 2018)
POS	
HIN-ENG	FastText: Hindi, English (Grave et al., 2018)
SPA-ENG	FastText: Spanish, English, Catalan, Portugese (Grave et al., 2018) GLoVe: English-Twitter (Pennington et al., 2014)

Table 2: Embeddings list for MME.

	Subword Embeddings List
NER	
HIN-ENG	Hindi, English
SPA-ENG	Spanish, English, Catalan, Portugese
MSA-EA	Arabic, Egyptian
POS	
HIN-ENG	Hindi, English
SPA-ENG	Spanish, English, Catalan, Portugese

Table 3: Subword embeddings list for HME.

3.3 Other Baselines

We compare the results with Char2subword and mBERT (cased) from Aguilar et al. (2020b). We also include the results of English BERT provided by the organizer of the LinCE public benchmark leaderboard (accessed on March 12nd, 2021). [3]

4 Results and Discussions

4.1 LinCE Benchmark

We evaluate all the models on the LinCE benchmark, and the development set results are shown in Table 4. As expected, models without any pretraining (e.g., Scratch (4L)) perform significantly worse than other pre-trained models. Both FastText and MME use pre-trained word embeddings, but MME achieves a consistently higher F1 score than FastText in both NER and POS tasks, demonstrating the importance of the contextualized self-attentive encoder. HME further improves on the F1 score of the MME models, suggesting that encoding hierarchical information from sub-word level, word level, and sentence level representations can improve code-switching task performance. Comparing HME with contextualized pre-trained mul-

[1] https://github.com/huggingface/transformers

[2] https://github.com/gentaiscool/meta-emb

[3] https://ritual.uh.edu/lince

		NER						POS			
		HIN-ENG		SPA-ENG		MSA-EA		HIN-ENG		SPA-ENG	
Method	**Avg Perf.**	Params	F1	Params	F1	Params	F1	Params	Acc	Params	Acc
Scratch (2L)	63.40	96M	46.51	96M	32.75	96M	60.14	96M	83.20	96M	94.39
Scratch (4L)	60.93	111M	47.01	111M	19.06	111M	60.24	111M	83.72	111M	94.64
Mono/Multilingual Word Embeddings											
FastText (ML)	76.43	4M	63.58	18M	57.10	16M	78.42	4M	84.63	6M	98.41
FastText (EL)	76.71	4M	69.79	18M	58.34	16M	72.68	4M	84.40	6M	98.36
MUSE (ML → EL)	76.54	4M	64.05	18M	58.00	16M	78.50	4M	83.82	6M	98.34
MUSE (EL → ML)	75.58	4M	64.86	18M	57.08	16M	73.95	4M	83.62	6M	98.38
Pre-Trained Multilingual Models											
mBERT (uncased)	79.46	167M	68.08	167M	63.73	167M	78.61	167M	90.42	167M	96.48
mBERT (cased)[‡]	79.97	177M	72.94	177M	62.66	177M	78.93	177M	87.86	177M	97.29
Char2Subword[‡]	81.07	136M	74.91	136M	63.32	136M	80.45	136M	89.64	136M	97.03
XLM-R$_{\text{BASE}}$	81.90	278M	76.85	278M	62.76	278M	81.24	278M	91.51	278M	97.12
XLM-R$_{\text{LARGE}}$	**84.39**	565M	**79.62**	565M	**67.18**	565M	**85.19**	565M	92.78	565M	97.20
XLM-MLM$_{\text{LARGE}}$	81.41	572M	73.91	572M	62.89	572M	82.72	572M	90.33	572M	97.19
Multilingual Meta-Embeddings											
Concat	79.70	10M	70.76	86M	61.65	31M	79.33	8M	88.14	23M	98.61
Linear	79.60	10M	69.68	86M	61.74	31M	79.42	8M	88.58	23M	98.58
Attention (MME)	79.86	10M	71.69	86M	61.23	31M	79.41	8M	88.34	23M	98.65
HME	81.60	12M	73.98	92M	62.09	35M	81.26	12M	92.01	30M	98.66
HME-Ensemble	**82.44**	20M	76.16	103M	62.80	43M	81.67	20M	**92.84**	40M	**98.74**

Table 4: Results on the development set of the LinCE benchmark. [‡] The results are taken from Aguilar et al. (2020b). The number of parameters of mBERT (cased) is calculated by approximation.

			NER			POS	
Method	**Avg Params**	**Avg Perf.↑**	HIN-ENG	SPA-ENG	MSA-EA	HIN-ENG	SPA-ENG
English BERT (cased)[†]	108M	75.80	74.46	61.15	59.44	87.02	96.92
mBERT (cased)[‡]	177M	77.08	72.57	64.05	65.39	86.30	97.07
HME	36M	77.64	73.78	63.06	66.14	88.55	96.66
Char2Subword[‡]	136M	77.85	73.38	64.65	66.13	88.23	96.88
XLM-MLM$_{\text{LARGE}}$	572M	78.40	74.49	64.16	67.22	89.10	97.04
XLM-R$_{\text{BASE}}$	278M	78.75	75.72	64.95	65.13	91.00	96.96
HME-Ensemble	45M	79.17	75.97	65.11	**68.71**	89.30	96.78
XLM-R$_{\text{LARGE}}$	565M	**80.96**	**80.70**	**69.55**	65.78	**91.59**	**97.18**

Table 5: Results on the test set of the LinCE benchmark.[‡] The results are taken from Aguilar et al. (2020b). [†] The result is taken from the LinCE leaderboard.

tilingual models such as mBERT and XLM-R, we find that HME models are able to obtain competitive F1 scores while maintaining a 10x smaller model sizes. This result indicates that pre-trained multilingual word embeddings can achieve a good balance between performance and model size in code-switching tasks. Table 5 shows the models' performance in the LinCE test set. The results are highly correlated to the results of the development set. XLM-R$_{\text{LARGE}}$ achieves the best-averaged performance, with a 13x larger model size compared to the HME-Ensemble model.

4.2 Model Effectiveness and Efficiency

Performance vs. Model Size As shown in Figure 2, the Scratch models yield the worst average score, at around 60.93 points. With the smallest pre-trained embedding model, FastText, the model performance can improve by around 10 points compared to the Scratch models and they only have 10M parameters on average. On the other hand, the MME models, which have 31.6M parameters on average, achieve similar results to the mBERT models, with around 170M parameters. Interestingly, adding subwords and character embeddings to MME, such as in the HME models, further im-

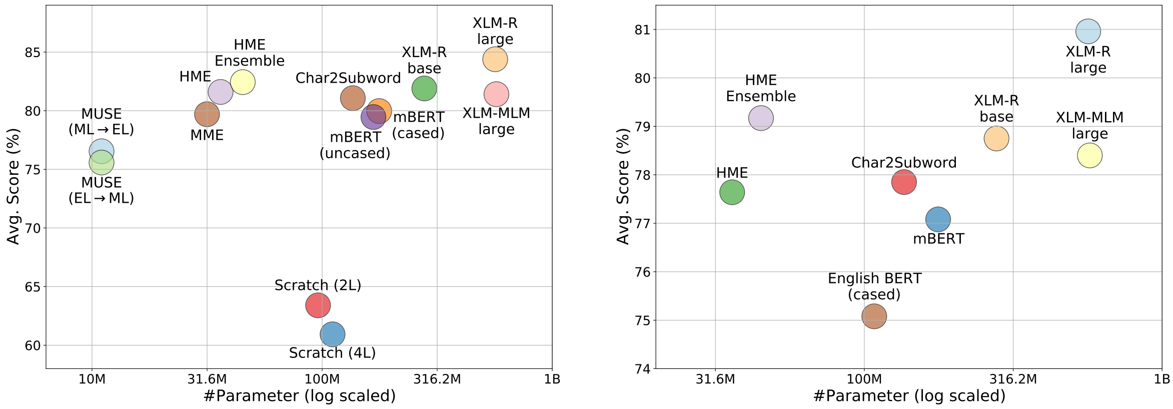


Figure 2: Validation set (left) and test set (right) evaluation performance (y-axis) and parameter (x-axis) of different models on LinCE benchmark.

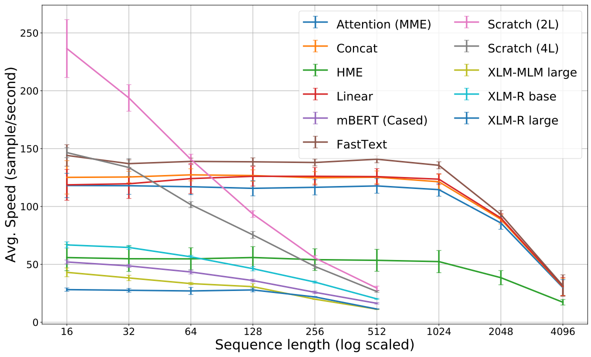


Figure 3: Speed-to-sequence length comparison of different models.

proves the performance of the MME models and achieves a 81.60 average score, similar to that of the XLM-R$_{\text{BASE}}$ and XLM-MLM$_{\text{LARGE}}$ models, but with less than one-fifth the number of parameters, at around 42.25M. The Ensemble method adds further performance improvement of around 1% with an additional 2.5M parameters compared to the non-Ensemble counterparts.

Inference Time To compare the speed of different models, we use generated dummy data with various sequence lengths, [16, 32, 64, 128, 256, 512, 1024, 2048, 4096]. We measure each model's inference time and collect the statistics of each model at one particular sequence length by running the model 100 times. The experiment is performed on a single NVIDIA GTX1080Ti GPU. We do not include the pre-processing time in our analysis. Still, it is clear that the pre-processing time for meta-embeddings models is longer than for other models as pre-processing requires a tokenization step to be conducted for the input multiple times with different tokenizers. The sequence lengths are counted based on the input tokens of each model. We use words for the MME and HME models, and subwords for other models.

The results of the inference speed test are shown in Figure 3. Although all pre-trained contextualized language models yield a very high validation score, these models are also the slowest in terms of inference time. For shorter sequences, the HME model performs as fast as the mBERT and XLM-R$_{\text{BASE}}$ models, but it can retain the speed as the sequence length increases because of the smaller model dimension in every layer. The FastText, MME, and Scratch models yield a high throughput in short-sequence settings by processing more than 150 samples per second. For longer sequences, the same behavior occurs, with the throughput of the Scratch models reducing as the sequence length increases, even becoming lower than that of the HME model when the sequence length is greater than or equal to 256. Interestingly, for the FastText, MME, and HME models, the throughput remains steady when the sequence length is less than 1024, and it starts to decrease afterwards.

Memory Footprint We record the memory footprint over different sequence lengths, and use the same setting for the FastText, MME, and HME models as in the inference time analysis. We record the size of each model on the GPU and the size of the activation after performing one forward operation to a single sample with a certain sequence length. The result of the memory footprint analysis for a sequence length of 512 is shown in Table 6. Based on the results, we can see that meta-embedding models use a significantly smaller memory footprint to store the model and activation memory. For instance, the memory footprint of the HME model is less than that of the Scratch (4L) model, which has only four transformer encoder layers, a model dimension of 768 and a feed-forward dimension of 3,072. On the other hand, large pre-trained language models, such as XLM-MLM$_{\text{LARGE}}$ and XLM-R$_{\text{LARGE}}$, use a much larger memory for storing the activation memory compared to all other models. The complete results of the memory footprint analysis are shown in Appendix A.

Model	Activation (MB)
FastText	79.0
Concat	85.3
Linear	80.8
Attention (MME)	88.0
HME	154.8
Scratch (2L)	133.0
Scratch (4L)	264.0
mBERT	597.0
XLM-R$_{\text{BASE}}$	597.0
XLM-R$_{\text{LARGE}}$	1541.0
XLM-MLM$_{\text{LARGE}}$	1158.0

Table 6: GPU memory consumption of different models with input size of 512.

5 Related Work

Transfer Learning on Code-Switching Previous works on code-switching have mostly focused on combining pre-trained word embeddings with trainable character embeddings to represent noisy mixed-language text (Trivedi et al., 2018; Wang et al., 2018b; Winata et al., 2018c). Winata et al. (2018a) presented a multi-task training framework to leverage part-of-speech information in a language model. Later, they introduced the MME in the code-switching domain by combining multiple word embeddings from different languages (Winata et al., 2019a). MME has since also been applied to Indian languages (Priyadharshini et al., 2020; Dowlagar and Mamidi, 2021).

Meta-embeddings have been previously explored in various monolingual NLP tasks (Yin and Schütze, 2016; Muromägi et al., 2017; Bollegala et al., 2018; Coates and Bollegala, 2018; Kiela et al., 2018). Winata et al. (2019b) introduced hierarchical meta-embeddings by leveraging subwords and characters to improve the code-switching text representation. Pratapa et al. (2018b) propose to train skip-gram embeddings from synthetic code-switched data generated by Pratapa et al. (2018a). This improves syntactic and semantic code-switching tasks. Winata et al. (2018b); Lee et al. (2019); Winata et al. (2019d); Samanta et al. (2019), and Gupta et al. (2020) proposed a generative-based model for augmenting code-switching data from parallel data. Recently, Aguilar et al. (2020b) proposed the Char2Subword model, which builds representations from characters out of the subword vocabulary, and they used the module to replace subword embeddings that are robust to misspellings and inflection that are mainly found in a social media text. Khanuja et al. (2020) explored fine-tuning techniques to improve mBERT for code-switching tasks, while Winata et al. (2020) introduced a meta-learning-based model to leverage monolingual data effectively in code-switching speech and language models.

Bilingual Embeddings In another line of works, bilingual embeddings have been introduced to represent code-switching sentences, such as in bilingual correlation-based embeddings (BiCCA) (Faruqui and Dyer, 2014), the bilingual compositional model (BiCVM) (Hermann and Blunsom, 2014), BiSkip (Luong et al., 2015), RCSLS (Joulin et al., 2018), and MUSE (Lample et al., 2017, 2018), to align words in L1 to the corresponding words in L2, and vice versa.

6 Conclusion

In this paper, we study multilingual language models' effectiveness so as to understand their capability and adaptability to the mixed-language setting. We conduct experiments on named entity recognition and part-of-speech tagging on various language pairs. We find that a pre-trained multilingual model does not necessarily guarantee high-quality representations on code-switching, while the hierarchical meta-embeddings (HME) model achieve similar results to mBERT and XLM-R$_{\text{BASE}}$ but with significantly fewer parameters. Interestingly, we find that XLM-R$_{\text{LARGE}}$ has better performance by a great margin, but with a substantial cost in the training and inference time, using 13x more parameters than HME-Ensemble for only a 2% improvement.

Acknowledgments

This work has been partially funded by ITF/319/16FP and MRP/055/18 of the Innovation Technology Commission, the Hong Kong

SAR Government, and School of Engineering Ph.D. Fellowship Award, the Hong Kong University of Science and Technology, and RDC 1718050-0 of EMOS.AI.

References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named entity recognition on code-switched data: Overview of the calcs 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147.

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

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López-Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 148–153.

Gustavo Aguilar, Bryan McCann, Tong Niu, Nazneen Rajani, Nitish Keskar, and Thamar Solorio. 2020b. Char2subword: Extending the subword embedding space from pre-trained models using robust character compositionality. *arXiv preprint arXiv:2010.12730*.

Danushka Bollegala, Kohei Hayashi, and Ken-Ichi Kawarabayashi. 2018. Think globally, embed locally: locally linear meta-embedding of words. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3970–3976. AAAI Press.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. 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 194–198.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard 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.

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.

Suman Dowlagar and Radhika Mamidi. 2021. Cmsaone@ dravidian-codemix-fire2020: A meta embedding and transformer model for code-mixed sentiment analysis on social media text. *arXiv preprint arXiv:2101.09004*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

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

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2267–2280.

Benjamin Heinzerling and Michael Strube. 2018. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 58–68, Baltimore, Maryland. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multitask benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.

Armand Joulin, Piotr Bojanowski, Tomáš Mikolov, Hervé Jégou, and Édouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. Gluecos: An evaluation benchmark for code-switched nlp. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477.

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Grandee Lee, Xianghu Yue, and Haizhou Li. 2019. Linguistically motivated parallel data augmentation for code-switch language modeling. In *INTERSPEECH*, pages 3730–3734.

Zhaojiang Lin, Zihan Liu, Genta Indra Winata, Samuel Cahyawijaya, Andrea Madotto, Yejin Bang, Etsuko Ishii, and Pascale Fung. 2020. Xpersona: Evaluating multilingual personalized chatbot. *arXiv preprint arXiv:2003.07568*.

Zihan Liu, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. 2020. Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8433–8440.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Denver, Colorado. Association for Computational Linguistics.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Avo Muromägi, Kairit Sirts, and Sven Laur. 2017. Linear ensembles of word embedding models. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 96–104.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018a. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1543–1553.

Adithya Pratapa, Monojit Choudhury, and Sunayana Sitaram. 2018b. Word embeddings for code-mixed language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3067–3072, Brussels, Belgium. Association for Computational Linguistics.

Ruba Priyadharshini, Bharathi Raja Chakravarthi, Mani Vegupatti, and John P McCrae. 2020. Named entity recognition for code-mixed indian corpus using meta embedding. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 68–72. IEEE.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code-switched text. *arXiv preprint arXiv:1906.08972*.

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

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018b. A twitter corpus for hindi-english code mixed pos tagging. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 12–17.

Victor Soto and Julia Hirschberg. 2017. Crowdsourcing universal part-of-speech tags for code-switching. *Proc. Interspeech 2017*, pages 77–81.

Shashwat Trivedi, Harsh Rangwani, and Anil Kumar Singh. 2018. Iit (bhu) submission for the acl shared task on named entity recognition on code-switched data. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 148–153.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Changhan Wang, Kyunghyun Cho, and Douwe Kiela. 2018b. Code-switched named entity recognition with embedding attention. In *Proceedings of the*

Third Workshop on Computational Approaches to Linguistic Code-Switching, pages 154–158.

Bryan Wilie, Karissa Vincentio, Genta Indra Winata, Samuel Cahyawijaya, Xiaohong Li, Zhi Yuan Lim, Sidik Soleman, Rahmad Mahendra, Pascale Fung, Syafri Bahar, et al. 2020. Indonlu: Benchmark and resources for evaluating indonesian natural language understanding. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 843–857.

Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, Peng Xu, and Pascale Fung. 2020. Meta-transfer learning for code-switched speech recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3770–3776.

Genta Indra Winata, Zhaojiang Lin, and Pascale Fung. 2019a. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186.

Genta Indra Winata, Zhaojiang Lin, Jamin Shin, Zihan Liu, and Pascale Fung. 2019b. Hierarchical meta-embeddings for code-switching named entity recognition. 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 3532–3538.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Jamin Shin, Yan Xu, Peng Xu, and Pascale Fung. 2019c. Caire_hkust at semeval-2019 task 3: Hierarchical attention for dialogue emotion classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 142–147.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018a. Code-switching language modeling using syntax-aware multi-task learning. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 62–67.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018b. Learn to code-switch: Data augmentation using copy mechanism on language modeling. *arXiv preprint arXiv:1810.10254*.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019d. Code-switched language models using neural based synthetic data from parallel sentences. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 271–280.

Genta Indra Winata, Chien-Sheng Wu, Andrea Madotto, and Pascale Fung. 2018c. Bilingual character representation for efficiently addressing out-of-vocabulary words in code-switching named entity recognition. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 110–114.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. 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 833–844.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1351–1360.

A Memory Footprint Analysis

We show the complete results of our memory footprint analysis in Table 7.

Model	**Activation (MB)**								
	16	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**	**4096**
FastText	1.0	2.0	4.0	10.0	26.0	79.0	261.0	941.0	3547.0
Linear	1.0	2.0	4.0	10.0	27.4	80.8	265.6	950.0	3562.0
Concat	1.0	2.0	5.0	11.2	29.2	85.2	274.5	967.5	3596.5
Attention (MME)	1.0	2.0	5.4	12.4	31.0	89.0	283.2	985.6	3630.6
HME	3.2	6.6	13.4	28.6	64.2	154.8	416.4	1252.0	4155.0
Scratch (2L)	2.0	4.0	8.0	20.0	46.0	133.0	-	-	-
Scratch (4L)	3.0	7.0	15.0	38.0	90.0	264.0	-	-	-
mBERT (uncased)	10.0	20.0	41.0	100.0	218.0	597.0	-	-	-
XLM-R$_{\text{BASE}}$	10.0	20.0	41.0	100.0	218.0	597.0	-	-	-
XLM-R$_{\text{LARGE}}$	25.0	52.0	109.0	241.0	579.0	1541.0	-	-	-
XLM-MLM$_{\text{LARGE}}$	20.0	42.0	89.0	193.0	467.0	1158.0	-	-	-

Table 7: Memory footprint (MB) for storing the activations for a given sequence length.