Contextual Embeddings for Arabic-English Code-Switched Data

Caroline Sabty, Mohamed Islam, and Slim Abdennadher
The German University in Cairo,
El Tagamoa El Khames, New Cairo, Cairo, Egypt
caroline.samy,slim.abdennadher@guc.edu.eg
mohamed.islam@student.guc.edu.eg

Abstract

Globalization has caused the rise of the code-switching phenomenon among multilingual societies. In Arab countries, code-switching between Arabic and English has become frequent, especially through social media platforms. Consequently, research in Natural Language Processing (NLP) systems increased to tackle such a phenomenon. One of the significant challenges of developing code-switched NLP systems is the lack of data itself. In this paper, we propose an open source trained bilingual contextual word embedding models of FLAIR, BERT, and ELECTRA. We also propose a novel contextual word embedding model called KERMIT, which can efficiently map Arabic and English words inside one vector space in terms of data usage. We applied intrinsic and extrinsic evaluation methods to compare the performance of the models. Our results show that FLAIR and FastText achieve the highest results in the sentiment analysis task. However, KERMIT is the best-achieving model on the intrinsic evaluation and named entity recognition. Also, it outperforms the other transformer-based models on question answering task.

1 Introduction

Nowadays, due to globalization, bilingual communities tend to code-switch in everyday communication. Code-Switching (CS) is defined as the embedding of linguistic units such as phrases, words, and morphemes of one language into an utterance of another language (Myers-Scotton, 1997). This linguistic behavior occurs on both spoken and written scales. Primary language appears the most inside an utterance, while secondary language is the language of embedded words or phrases inside an utterance. The phenomenon of code-switching has been increasingly reported in linguistic studies in the past years as more people tend to code-switch. Furthermore, this phenomenon has become popular in Arab countries, where people code-switch between different dialects or their dialect and foreign languages. For instance, it is common to mix between Arabic and French in Tunisia and Egyptian Arabic and English in Egypt. CS behavior is common in online interaction, especially among social media users, generating vast amounts of CS data (Barman et al., 2014). For example, on Twitter, the multilingual users are more active than the monolingual ones (Hale, 2014). Thus, it is essential nowadays to process and understand the natural language of humans. This is performed by applying several Natural Language Processing (NLP) techniques while using different deep learning approaches. Lately, CS data started to gain attention in different NLP tasks (Hamed et al., 2017; Menacer et al., 2019; Sabty et al., 2019a).

One of the powerful developments in the NLP field is word embeddings; they represent words as vectors in a continuous space. This leads to having semantically similar words grouped near each other. It has been proven that adding word embedding instead of the traditional bag-of-words approach to different NLP tasks is very efficient (Soliman et al., 2017). Recently, bilingual word embeddings gained much attention to embedding in the same space words from two languages. Most of the standard bilingual word embedding techniques are intended to be trained and work on monolingual texts, not on a mix of two languages. Thus, they are not the ideal option to learn embeddings for code-switched tasks (Pratapa et al., 2018b).

As our focus in this work is on the Arabic-English CS data, previously (Hamed et al., 2019) developed a classical word embedding model called Bi-CS. This model produced classical word embeddings by training on CS Arabic-English corpus. It was trained on monolingual data and a small amount of CS data, which acted as a gluing force bringing the monolingual embeddings closer in the vector space. However, classical word embeddings compute static vectors for each word. In polysemous words that depend on the context, classical word embeddings fail to model these words. Contextual embeddings, on the other hand, compute context-dependent nature vectors for words. This work aims to train bilingual contextual embedding models using state-of-the-art embedding types generated from our collected code-switched Arabic-English corpus. The models we created were using FLAIR, BERT, and ELECTRA. We also proposed a new contextual word embedding model called KERMIT based on the previous work (Devlin et al., 2018; Clark et al., 2020), capable of mapping both Arabic and English words inside one vector space efficiently in terms of data usage. All our trained and proposed models are available as an open source[1].

To compare our bilingual embedding models, we applied intrinsic and extrinsic evaluation methods. In the intrinsic evaluation, the models are tested on their ability to assign near-by vectors to similar tokens. This was achieved by calculating the cosine similarity between the tokens using BERTS$_{\text{CORE}}$ (Zhang et al., 2019). In the extrinsic evaluation, we evaluated our models on three downstream NLP tasks; Named Entity Recognition, Sentiment Analysis, and Question Answering on Arabic-English CS text. As a result of the intrinsic evaluation, the best model is KERMIT, which is capable of modeling higher cosine-similarity with similar code-switched words and lower for unrelated ones. In addition, the system we implemented using KERMIT is the best-achieving model for the NER task on Arabic-English data. Regarding the sentiment analysis, FLAIR and FastText achieved higher results compared to all other models. This shows that the character embedding outperformed the different types of embedding in some tasks on CS data. Results also show that the KERMIT model achieved the highest results on question answering task across the transformer-based models.

The paper is organized as follows. Section 2 presents some related work for word embedding models. In Section 3, the process of data collection and corpus creation is illustrated. Section 4 explains the different types of embedding models we trained. Section 5 discusses the different evaluation methods and their results. Finally, Section 7 concludes the paper and presents future work.

2 Related Work

Some studies implemented different word embedding models to deal with various Arabic NLP tasks. For example, Soliman et al. (2017) implemented Arabic pre-trained word embedding models. Their first version contained six distinct models built using either continuous Bag-of-Words (CBOW) or Skip-Gram (SG) techniques for the different text domains. They measured the similarity of word vectors of a subset of sentiment words and named entities to evaluate their models. Then they applied the clustering technique to see if, in each set, the words with the same polarity will be clustered together or no. They also used SemEval-2017 Semantic Textual Similarity to check how equivalent paired snippets of text. Another set of word embedding models for the Arabic language is presented in (Fouad et al., 2020). The models are implemented using CBOW, SG, and GloVe techniques, and they are generated from a set of Arabic tweets. They proposed a new way to measure the similarity of the Arabic words to evaluate the performance of their proposed models. Besides, they tested the performance in the multi-class sentiment analysis classification task.

Bilingual models have been explored as a link to bridge the gap between languages in CS word embedding and building models for low-resource languages. Several bilingual models have been proposed to align cross-lingual data, they used different alignment techniques such as word-level (Hermann and Blunsom, 2014; Faruqui and Dyer, 2014), sentence-level (Hermann and Blunsom, 2014; Gouws et al., 2015), both word and sentence level (Luong et al., 2015) and document-level alignments (Vulic and Moens,

[1]https://github.com/CSabty/Code-Switch-Arabic-English-Contextual-Embeddings

2015; Vulić and Moens, 2016). Besides Upadhyay et al. (2016) presented a comparison between four different cross-lingual embeddings models (Luong et al., 2015; Hermann and Blunsom, 2014; Faruqui and Dyer, 2014; Vulic and Moens, 2015), they vary in terms of the amount of supervision. Pratapa et al. (2018b) compared between the three bilingual models (Hermann and Blunsom, 2014; Faruqui and Dyer, 2014; Luong et al., 2015) for enhancing the downstream tasks of sentiment analysis and POS tagging on English-Spanish CS data. Moreover, they proposed an approach for training CS text generated by (Pratapa et al., 2018a) using skip-grams. In (Lachraf et al., 2019), they proposed several Arabic-English cross-lingual word embedding models trained on pairs of Arabic-English parallel sentences. It is essential to mention that training cross-lingual and multilingual embeddings require monolingual data. The set of syntactic structures and semantic associations of code-switched text are not shown in monolingual sentences. Thus, learning CS text analysis using cross-lingual and multilingual embeddings is not optimal. Code-switched embedding should be trained using CS text (Pratapa et al., 2018b).

Gao et al. (2019) proposed an approach to employ the BERT model and Generative Adversarial Net model for CS text generation. The developed system is capable of generating full CS sentences by training on low resourced CS data. Through this process, the BERT model learns contextual embedding representation. They evaluated their generated data in an ASR system on Mandarin-English CS data. A Multi-Encoder-Decoder Transformer model was proposed in (Zhou et al., 2020) for CS data. This model involves two language-specific encoder modules, each trained on monolingual data. These two modules are then integrated and trained on low resourced CS data. Eventually, this module is capable of representing contextual embedding.

Related to our work for Arabic-English CS data, Hamed et al. (2019) compared different bilingual embeddings (Hermann and Blunsom, 2014; Faruqui and Dyer, 2014; Luong et al., 2015) having different cross-lingual supervision. They also proposed two extensions, one of which depends on monolingual and small CS corpora, and another one combines the first two approaches. They evaluated the effect of using different embeddings in language modeling. However, the proposed embeddings are classical ones and do not consider the context of the words.

3 Data Collection

To train the word embedding models using Arabic-English code-switched text, we collected the data using three different techniques/sources and created two corpora. The first corpus *CS_TRAIN* is composed of 105 million tokens. The first source of data was using the CS corpus of (Hamed et al., 2019) collected from social media platforms. The second technique is generating 30 million Arabic-English CS tokens by translating monolingual Modern Standard Arabic corpus into Arabic-English CS data. We iterated automatically over the corpus and translated tokens according to a set of linguistic constraints using an open-sourced Neural Machine Translation API[2]. We followed the linguistic constraints inferred from evaluating real Arabic-English code-switched data in (Hamed et al., 2018). They defined trigger words such as (ال, the), (في, in) and (و, and), which are Arabic words preceding a CS point. The final part of the corpus is composed from the monolingual Arabic news-wire data. To augment the size of the training data, we created another corpus *CS_TRAIN*$_{++}$ by adding to the initial one *CS_TRAIN* 20 million tokens from the same Arabic monolingual news-wire data-set. Also, we translated them following the same linguistic constraints to form the other 19 million CS tokens. The corpus of *CS_TRAIN*$_{++}$ is composed of 144 million tokens. The corpora statistics are presented in Table 1.

Data-set	English Tokens	Arabic Tokens	Sentences
CS_TRAIN	10M	95M	7M
CS_TRAIN$_{++}$	17M	127M	9M

Table 1: Detailed data statistics about the number of English tokens, Arabic tokens, the number of sentences

[2]https://rapidapi.com/gofitech/api/nlp-translation

4 Embedding Models

We trained several bilingual contextual embedding models to compare them and have an efficient model for Arabic-English CS data used in several NLP tasks. We used several state-of-the-art techniques and the type of contextual embeddings for building our models. We started by creating a baseline model using a stack of Arabic pre-trained Pooled FLAIR embedding and pre-trained FastText. The other models we built are using FLAIR, BERT, and ELECTRA. We also proposed a new model called the KERMIT.

4.1 Baseline

The baseline model used in our experiment is a stack of Arabic pre-trained Pooled FLAIR embedding (Akbik et al., 2019b) and pre-trained FastText. Pooled FLAIR embedding and FastText achieved the best results on NER downstream task for Arabic-English CS data in (Sabty et al., 2019b). It even outperformed the only available Arabic-English CS embedding of Bi-CS (Hamed et al., 2019). Pooling operation dynamically aggregates each unique word encountered, then retrieves previous embeddings produced from memory. Finally, the pool operation is performed, and all locally contextualized embeddings are concatenated to make the final embedding for the token. FastText can compute classical word embedding representation using the same methodology and architecture as the word2vec model (Mikolov et al., 2013) with an extension. FastText encodes character-level representations for a word, it breaks words into n-grams, and the word embedding is computed as the sum of all these n-grams. This capability helps FastText in outperforming word2vec in modeling word representations.

4.2 FLAIR

The first model trained is FLAIR (Akbik et al., 2018), a contextualized character-level language model. FLAIR model sentences as a sequence of characters and trains on the auto-regressive task. FLAIR can model sequence bidirectionally by stacking a forward and a backward character language model. Therefore, to get the whole context, we have pre-trained forward and backward FLAIR language models. FLAIR models can produce several embeddings for the same word based on its context and manage dealing with rare and misspelled words by modeling context as characters.

The pre-processing stage involved removing punctuation symbols and lower casing English characters in the corpus. In the tokenization phase, we have listed all the alphabetical English and Arabic characters. Hyper-parameters were configured following the original FLAIR parameters (Akbik et al., 2018). Forward and Backward language models were trained using the same vocabulary. We tracked the training performance on the validation set. Training took ten days for each of the backward and forward models on one GPU or halted when negligible gains were observed.

4.3 BERT

We have trained BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) on our CS corpora. BERT is a transformer-based language model. It is trained on the Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks. MLMs are trained on masking out random tokens by replacing them with special token [MASK] inside a sentence. The model then tries to predict the masked tokens using the whole context of a sequence. Also, NSP models are trained on distinguishing whether two input sentences are continuous segment or separate. Unlike FLAIR, BERT models sentences as a sequence of tokens. Each input token in a sequence flows through stacked encoders and outputs a hidden state representing the word embedding. This enables BERT to train in parallel.

We have implemented an additional pre-processing stage to our data set before proceeding to the training phase. This stage involves segmenting Arabic words using Farasa segmenter (Abdelali et al., 2016) to remove redundant forms of words. After segmenting the corpus, we produced a total of 64k tokens as a vocabulary for our model. We have trained $\text{BERT}_{\text{BASE}}$ sized model with 12 encoder layers, a hidden size of 768, and 12 attention heads. Training BERT is done by masking 15% of tokens in the input, of which 80% are replaced with a special token [MASK], 10% are replaced with random tokens, and 10% remains as an original token. This configuration helps to prevent pretrain-finetune-discrepancy. A learning rate of 2e-5 and Adam optimizer was used in the pre-training. We trained our model for a total of 1,000,000 steps with a batch size of 128. The next 250,000 steps were then trained with a batch size of 256 to speed up the training process. The training took three days to complete on eight cloud TPUs.

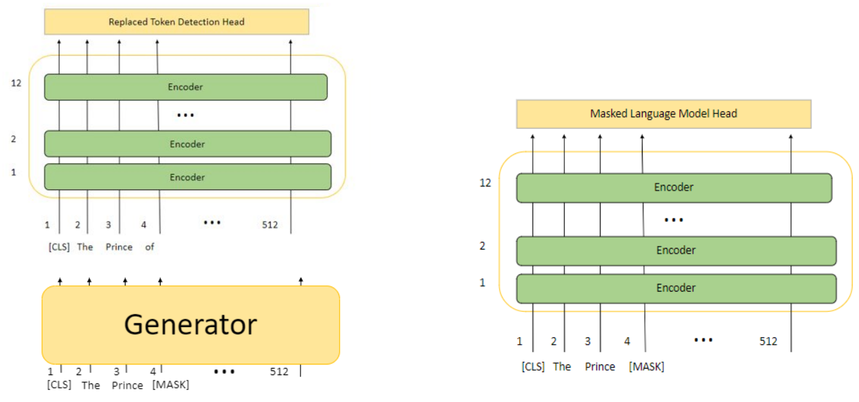
4.4 ELECTRA

We trained ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) (Clark et al., 2020) on our CS corpora. ELECTRA is composed of two neural networks, a generator and a discriminator trained together on MLM and Replaced Token Detection (RTD) tasks. RTD is a special unsupervised task that trained discriminative models. The language model is given a sequence of tokens from a generator and tries to predict whether a generator or original token replaces a token. A full modeling context does this. The ELECTRA model is trained to minimize the combined losses of the generator and the discriminator. Compared with BERT, the ELECTRA training mechanism is considered more efficient because the task is defined over all input tokens rather than just the 15% tokens that were masked out.

The pre-training stage was similar to BERT, however, with different configurations. We used the same pre-processing and tokenization mechanism. We have trained ELECTRA$_{\text{BASE}}$ sized model. The discriminator of this model has the same size as BERT$_{\text{BASE}}$ model. It is composed of 12 encoder layers, a hidden size of 128, 12 attention heads, and outputs embedding of size 768. The generator component of the ELECTRA$_{\text{BASE}}$ model is 1/3 of the size of the discriminator model. This configuration makes it harder for the discriminator component to distinguish tokens and produces more robust representations. We have trained both the generator and discriminator jointly from scratch with a learning rate of 2e-4 and batch size of 256 for a total of 750,000 steps. This training took five days on eigh TPUs.

4.5 KERMIT

We proposed a novel model called KERMIT for producing word embeddings. The architecture of this model is an encoder variant of transformers. The pre-training of this model is divided into two stages, as shown in Figure 1. In the first stage, KERMIT is trained as a discriminator in ELECTRA architecture using RTD and MLM tasks. After pre-training, the generator is dropped, and the pre-trained discriminator weights are used for the next stage. In the second stage, we initialize the encoder and embedding layers of the BERT model with the trained discriminator model. Then, we further trained the model as a generator using the MLM task. This training mechanism helps avoid pretrain-finetune-discrepancy and allows the model to learn stronger representation through training on more than one task.



(a) ELECTRA model

(b) BERT model

Figure 1: KERMIT encoder layers trained as discriminator in ELECTRA then as Generator in BERT

Furthermore, the model is trained on data with different augmentations. A generator replaced tokens at the first stage, and during the second stage, tokens were masked out. This helped to increase the diversity of data available for training. In the two stages of pre-training, we have used the same data pre-processing and tokenization vocabulary. We have trained the ELECTRA$_{\text{BASE}}$ model using the same hyper-parameters used for training the previous ELECTRA model. Using the same vocabulary, we transferred weights of the encoder and embedding layers and trained BERT$_{\text{BASE}}$ model using the same hyper-parameters used for training the previous BERT model. We have investigated different configurations for the transferred model. On the one hand, we trained the BERT model using trained generator prediction layers. On the other hand, we trained the model using randomly initialized generator prediction layers. Furthermore, we have changed the Adam optimizer variables so that loss affects only the encoder layers and applies less effect on the output layers. We have also tried to switch training stages through training BERT first then transferring weights to the ELECTRA model. The best configuration observed while training was using both randomly initialized generator prediction layers along with Adam optimizer variables.

5 Evaluation & Results

To evaluate our models, we have used intrinsic and extrinsic methods. We first trained the models using the *CS_TRAIN* and then using *CS_TRAIN*$_{++}$. For clarity, all models with a suffix ++ have been trained using *CS_TRAIN*$_{++}$.

5.1 Intrinsic Evaluation

The purpose of the intrinsic evaluation was to visualize and test how well the word embedding model can capture similarities in CS Arabic-English text. Thus, to evaluate our embedding models, we have leveraged cosine similarity between different tokens using BERTS$_{\text{CORE}}$ (Zhang et al., 2019). It is a new automatic evaluation metric for the generated text that computes token similarity using contextual embeddings. We evaluated the similarity between the tokens of several sentences, and the models performed the same in almost all of them. The following is an example of two CS sentences used to test the similarities stated by the models:

احمد يحب قراءه news كل day
'Ahmed loves reading news everyday' (a)

Mohamed ينهي كتاب في أسبوع
'Mohamed finishes a book in a week' (b)

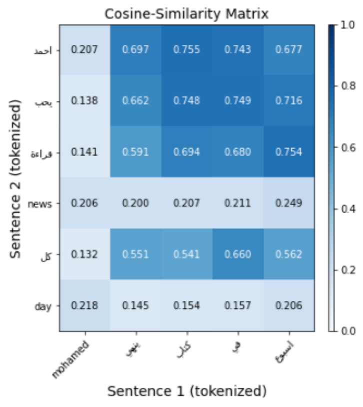
Examining the results in Figure 2, we can see how word embeddings can help state the similarity between CS words. All the models trained with *CS_TRAIN*$_{++}$ performed better than those trained with the smaller data-set *CS_TRAIN*. We also noted that the baseline model did not get the similarities of the English words. It is an expected result as it was trained on monolingual Arabic data. On the one hand, the similarities of the FLAIR model between all words is low. On the other hand, the similarities obtained by the ELECTRA model is high. BERT$_{++}$ is capable of capturing small similarities between tokens, including cross-lingual. Moreover, KERMIT$_{++}$ is the best one; it was capable of modeling higher cosine-similarity with similar words and lower for unrelated ones.

5.2 Extrinsic Evaluation

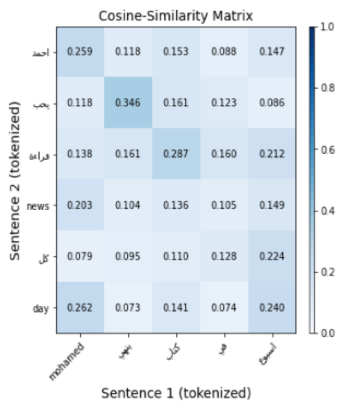
The extrinsic evaluation is based on the possibility of using the word embeddings in downstream NLP tasks. We have evaluated the impact of our models in three tasks; Named Entity Recognition, Sentiment Analysis and Question Answering on Arabic-English CS text. As to the best of our knowledge there is no available contextual embeddings trained on Arabic-English CS data, we evaluated the recent Arabic model AraBERT (Antoun et al., 2020) in the three tasks to compare its performance with our models.

5.2.1 Named Entity Recognition

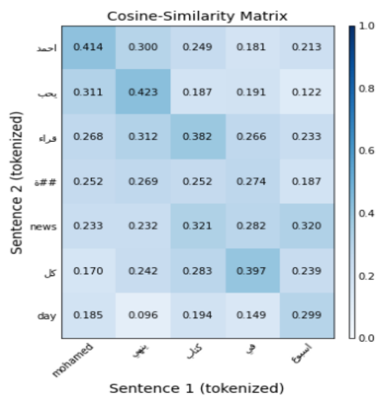
Named Entity Recognition (NER) is one of the essential tasks for NLP. It identifies named entities and classifies them into their types, e.g., person, location, and organization. We used two approaches to apply this task. The first one uses the deep learning NER model on Arabic-English CS data of (Sabty et al., 2019b), which is based on Bidirectional Long-Short-Term-Memory and Conditional Random Field along with the pre-trained word embedding layer. They compared the performance of their NER system using several types of embeddings. The best one was using Pooled FLAIR and FastText pre-trained Arabic



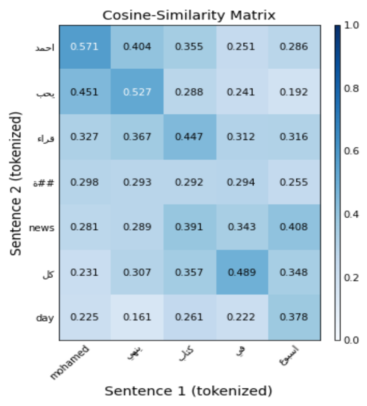
(a) Pooled FLAIR and FastText



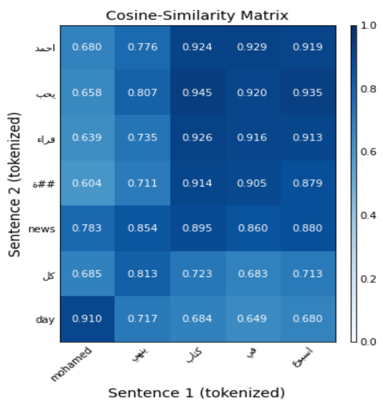
(b) FLAIR and FastText



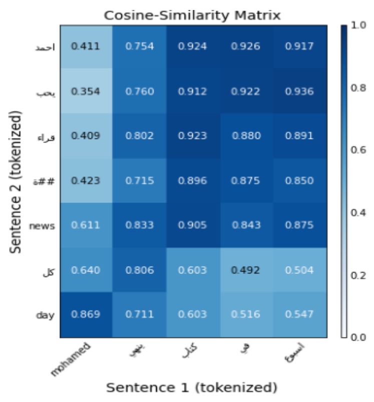
(c) BERT



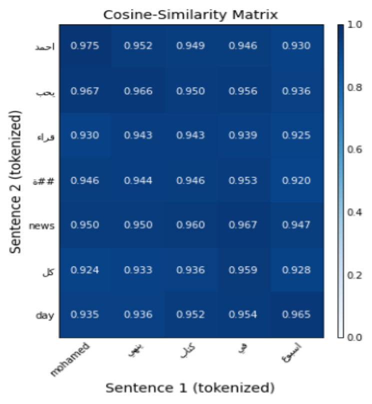
(d) BERT_{++}



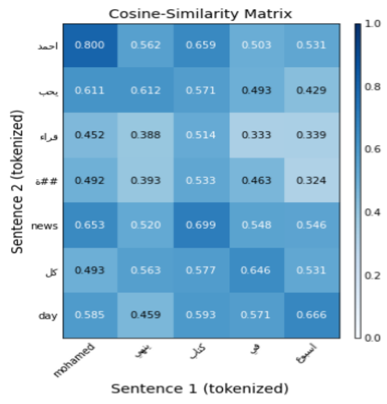
(e) ELECTRA



(f) ELECTRA_{++}



(g) KERMIT



(h) KERMIT_{++}

Figure 2: Results of comparing the tokens of two sentences using cosine-similarity of $\text{BERTS}_{\text{CORE}}$

embeddings, which we replicated in our baseline. Their best performance achieved an F-score equal to 77.69%. We used the same model and their training and testing data to compare the effect of using our different embeddings models on the performance. The second approach uses the NER system presented by the HuggingFace library that is based on fine-tuning (Wolf et al., 2019). We tested this system using the same training and testing of data of (Sabty et al., 2019b).

As shown in Table 2, we used all the models one by one as the embedding layer in the NER system, and we also combined some models. We combined FLAIR with Arabic FastText embedding, and it achieved the best results and enhanced the NER model by 0.51%. The results of using BERT_{++}, ELECTRA_{++}, and KERMIT_{++} are all less than the baseline model, but the three are almost the same with minimal differences. We combined the highest one of them KERMIT_{++} with the highest model of FLAIR and

Embedding Model	NER Model (Sabty et al., 2019b)	NER HuggingFace (Wolf et al., 2019)
Pooled FLAIR and FastText	77.69	-
FLAIR	72.55	-
FLAIR and FastText	**78.2**	-
KERMIT_{++} and FLAIR and FastText	76.6	-
AraBERT(Antoun et al., 2020)	58	66.7
BERT	64.5	76.5
BERT_{++}	68.2	77.1
ELECTRA	67	75.4
ELECTRA_{++}	68.3	76.29
KERMIT	65	78.7
KERMIT_{++}	69	**79.4**

Table 2: The F1-score results (%) of using the different types of embeddings in the two NER systems

FastText, it improved the results but still less than the baseline. However, in the HuggingFace system it was hard to add the Pooled FLAIR and FLAIR embeddings to their fine-tuning model since the pooling is over past embeddings. Thus, it is hard to fine tune all of them due to memory limitations. As a result KERMIT_{++} outperformed all other transformer-based models and even achieved higher results than the ones presented by the first NER approach by 1.2%.

5.2.2 Sentiment Analysis

We have also evaluated our models on sentiment analysis (SA) tasks using the methods proposed. It is a text classification task, where parts of the text are marked with binary labels to represent the sentiment type positive or negative in the text. We have created our data-set by translating a monolingual Arabic data-set of book reviews (Aly and Atiya, 2013) to have CS text using the same machine translation API and the linguistic constraints described previously in the data collection section. The results of experimenting sentiment analysis are stated in Table 3.

We used two approaches to apply this task. The first one uses the framework of FLAIR (Akbik et al., 2019a). It is based on the Recurrent Neural Network and linear classifier layers. While we added again to the FLAIR embedding the classical pre-trained FastText embedding, it improved the performance of the sentiment analysis model and scored the highest F1-score of 88.8%. The second uses the sentiment analysis system presented by HuggingFace based on fine-tuning (Linear Classifier) (Wolf et al., 2019); all transformer-based models are fine-tuned by adding a linear classifier layer. BERT model outperformed ELECTRA by about 1.6% and KERMIT by 0.9%. Furthermore BERT_{++} scored higher than ELECTRA_{++} by 0.4%. Finally, KERMIT_{++} scored the highest in the second approach. However, using the first approach with FLAIR and FastText outperformed all models experimented.

5.2.3 Question Answering

Another important task that we tested the effect of using our embeddings models with it is the question-answering. It is finding the part of the text in a context that answers a given question. We have created our own data-set by translating text from the questions of the monolingual Arabic Question Answering (QA) corpus presented by (Mozannar et al., 2019) following the same approach stated in the sentiment analysis task. We could not translate words from the answers, as this would lead to changing their order, which could be misleading for the question-answering task.

We used only one approach for this task, which uses the question answering system of HuggingFace, which is based on fine-tuning (Linear Classifier) (Wolf et al., 2019). As shown in Table 4 KERMIT_{++} achieved the highest score compared to all other fine-tuned models with an F1-score equal to 39.9%. It outperforms all models because, as state before training KERMIT as both generator and discriminator helps the model to compute better representation of the data. All the results of using the different embeddings are low as the answers in the training data did not contain CS behaviour, which could be enhanced

Embedding Model	SA HuggingFace (Wolf et al., 2019)	SA FLAIR Framework (Akbik et al., 2019a)
Pooled FLAIR and FastText	-	87.84
FLAIR	-	87.8
FLAIR and FastText	-	**88.8**
AraBERT(Antoun et al., 2020)	78.2	-
BERT	77.7	-
BERT_{++}	77.38	-
ELECTRA	76.39	-
ELECTRA_{++}	77.18	-
KERMIT	76.8	-
KERMIT_{++}	**79.9**	-

Table 3: The F1-score results (%) of using the different types of embeddings in the two sentiment analysis systems

using a more accurate translation technique without affecting the order of words. As shown in the results our models achieved higher results than AraBERT embedding in all the three tasks. This is due to the fact that AraBert is trained on monolingual Arabic data.

Embedding Model	QA HuggingFace (Wolf et al., 2019)
AraBERT(Antoun et al., 2020)	30
BERT	37.8
BERT_{++}	38.12
ELECTRA	34.1
ELECTRA_{++}	37.2
KERMIT	32.8
KERMIT_{++}	**39.9**

Table 4: The F1-score results (%) of using the different types of embeddings in the question answering system

6 Conclusion & Future Work

Word embedding is one of the cores of all NLP tasks. Pre-training word embeddings on extensive unlabelled data and using it on downstream tasks can save a lot of data annotation. However, it is a problem to find a large data-set when it comes to low resourced languages such as code-switched Arabic-English data. In this paper, we propose a solution to create a large corpus for pre-training word embeddings through several sources. We used this corpus for training different bilingual contextual embedding models: FLAIR, BERT, and ELECTRA. We also proposed a new model KERMIT. This model is trained as both a discriminator and a generator where each task has a different variant of the input. The different variants of the data helped the KERMIT model to train efficiently on the collected corpus. We have pre-trained and evaluated all the models on three downstream tasks. The character based-model FLAIR, along with FastText, outperformed all models in the task of sentiment analysis. However, KERMIT got better results in the NER task and outperformed the other transformer-based models in the question-answering task. Besides, we evaluated the performance of the models by calculating the cosine similarities between different tokens, and KERMIT captured the highest similarities.

In the future, more comparisons with available models could be done. We can pre-train variants of the model by keeping the monolingual data without CS text and compare the performance in different tasks. The models will be evaluated in different tasks using various annotated data-sets. An evaluation technique should be applied on the generated CS data to check how it simulates the CS behaviour.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 11–16.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019a. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019b. Pooled contextualized embeddings for named entity recognition. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 724–728.

Mohamed Aly and Amir Atiya. 2013. Labr: A large scale arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 494–498.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, page 9.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the first workshop on computational approaches to code switching*, pages 13–23.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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.

Mohammed M Fouad, Ahmed Mahany, Naif Aljohani, Rabeeh Ayaz Abbasi, and Saeed-Ul Hassan. 2020. Arwordvec: efficient word embedding models for arabic tweets. *Soft Computing*, 24(11):8061–8068.

Yingying Gao, Junlan Feng, Ying Liu, Leijing Hou, Xin Pan, and Yong Ma. 2019. Code-switching sentence generation by bert and generative adversarial networks. In *INTERSPEECH*, pages 3525–3529.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 748–756.

Scott A Hale. 2014. Global connectivity and multilinguals in the twitter network. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 833–842.

Injy Hamed, Mohamed Elmahdy, and Slim Abdennadher. 2017. Building a first language model for code-switch arabic-english. *Procedia Computer Science*, 117:208–216.

Injy Hamed, Mohamed Elmahdy, and Slim Abdennadher. 2018. Collection and analysis of code-switch egyptian arabic-english speech corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Injy Hamed, Moritz Zhu, Mohamed Elmahdy, Slim Abdennadher, and Ngoc Thang Vu. 2019. Code-switching language modeling with bilingual word embeddings: A case study for egyptian arabic-english. In *International Conference on Speech and Computer*, pages 160–170. Springer.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*.

Raki Lachraf, Youcef Ayachi, Ahmed Abdelali, Didier Schwab, et al. 2019. Arbengvec: Arabic-english cross-lingual word embedding model. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 40–48.

Minh-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.

Mohamed Amine Menacer, David Langlois, Denis Jouvet, Dominique Fohr, Odile Mella, and Kamel Smaïli. 2019. Machine translation on a parallel code-switched corpus. In *Canadian Conference on Artificial Intelligence*, pages 426–432. Springer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Hussein Mozannar, Karl El Hajal, Elie Maamary, and Hazem Hajj. 2019. Neural arabic question answering. *arXiv preprint arXiv:1906.05394*.

Carol Myers-Scotton. 1997. *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press.

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)*, 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.

Caroline Sabty, Mohamed Elmahdy, and Slim Abdennadher. 2019a. Named entity recognition on arabic-english code-mixed data. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 93–97. IEEE.

Caroline Sabty, Ahmed Sherif, Mohamed Elmahdy, and Slim Abdennadher. 2019b. Techniques for named entity recognition on arabic-english code-mixed data. *International Journal of Transdisciplinary AI*, 1(1):44–63.

Abu Bakr Soliman, Kareem Eissa, and Samhaa R El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *arXiv preprint arXiv:1604.00425*.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, volume 2, pages 719–725. ACL; East Stroudsburg, PA.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Xinyuan Zhou, Emre Yılmaz, Yanhua Long, Yijie Li, and Haizhou Li. 2020. Multi-encoder-decoder transformer for code-switching speech recognition. *arXiv preprint arXiv:2006.10414*.