Neural Bilingual Named Entity Recognizer for English - Spanish Code Switching

An abridged report for Computational Linguistics, WiSe 2023/24

Niranjana Hegde Bhimanakone Satyanarayana, 7057809

INTRODUCTION

Code Switching (CS) is a phenomenon that encompasses a mixture of different languages in a corpus or in a discourse. When people are comfortable with bilingualism or multilingualism, they tend to convey their ideas and have a natural conversation with usage of more than one language words (or phrases). With globalization, it is common for most of the world to be bilingual and get themselves exposed to the usage of different languages for different settings of daily life. However, people mix these languages at times to better convey their thoughts. With the rise in the popularity of social media platforms, the frequency with which code switching happens has also increased significantly over the last decade. It is also common across language classrooms where new learners of a language are often taught with a mix of different languages.

Switching between sentences is called “inter-sentential” and switching within the sentence is called “intra-sentential” [1]. However, there are different kinds of CS like tag switching [2] and intra-word switching [3]. This introduces new computational challenges in terms of parsing, tagging and also imposes linguistic difficulties in terms of semantic and pragmatic understanding of the sentence in question. Throughout the report, we do not differentiate between these two types and call them broadly as CS.

A named entity is any word or a phrase (usually a proper noun) that clearly identifies one item from a group of other items having similar or same characteristics. Some examples of named entities are the name of the person, organization, city, location, date time, currencies, events, etc. Named entity recognition (NER) is the process of identifying these named entities in the corpus/dataset and classifying them into their corresponding main categories.

Identifying the named entities is very crucial in many computational tasks. NER acts as an important preprocessing step for a plethora of downstream tasks like question answering, information retrieval, machine translation, etc. [4]. NER tasks are also necessary for the learning strategies like transfer learning, reinforcement learning and adversarial learning [4].

Commercially, NER is used for customer support, search engines, text autocomplete, recommendation systems. CS adds an additional complexity to these tasks because any named entity can have a different term/word in different languages. Any system that can effectively handle CS user inputs tends to have more user interaction and thus, in the commercial sense, generate more leads and revenue for the organizations.

The motive of the project is to thus explore the process of NER in the context of CS data using the novel application of neural approaches. We use two techniques to train the model i.e., one with the original dataset from CALCS 2018 Shared Task and second with the augmented data generated out of the same dataset. We compare and contrast the results obtained and discuss the observations.

DATASET

The dataset used is CALCS 2018 Shared Task: Spanish - English (SPA - ENG) data instance. This dataset is available in both Hugging Face [5] and Linguistic Code-switching Evaluation (LinCE) website [6]. This dataset was introduced during the 2018 CALCS competition for NER.

It consists of 67,223 tweets with 808,663 tokens. The train dataset contains 33,611 tweets, validation dataset contains 10,085 tweets and test dataset contains 23,527 tweets [7]. We count the total number of tweets in the train dataset that have words English alone, that have Spanish words alone and find the count to be 1,232 and 2,313 respectively.

The columns of the dataset are “words”, “ner”, and “lid”. The column “words” contains a list of all words corresponding to a tweet. The column “ner” refers to the list of NER labels associated with the words from “words” list and “lid” refers to the language of each word.

In “lid”, English is encoded as “lang1”, Spanish is encoded as “lang2” and there are other values like "ne", "fw", "ambiguous", "mixed", "other", "unk". “ne” stands for named entities which are of our primary interest. The above count was obtained by filtering the dataset where the lid list contains either “lang1” or “lang2” along with any other tags. The Code-switched tweets in the train dataset was found to be 8,692 where the filter logic involved finding a sentence whose lid list has both “lang1” and “lang2” (irrespective other above-mentioned IDs).

The target label class has 19 different labels and each corresponding NER label for a word is encoded within the list. They are:

- O
- B-PER

- I-PER
- B-LOC
- I-LOC
- B-ORG
- I-ORG
- B-PROD
- I-PROD
- B-EVENT
- I-EVENT
- B-GROUP
- I-GROUP
- B-TITLE
- I-TITLE
- B-TIME
- I-TIME
- B-OTHER
- I-OTHER

The labels are annotated using the IOB scheme to identify multiple words as a single named entity. B- prefix indicates the beginning of the entity and I- prefix indicates inside the same entity, following a B- tag. For example, if the sentence contains “John Doe”, then “John” is marked as “B-PER” and “Doe” is marked as “I-PER”. PER refers to Person, LOC refers to Location, ORG refers to Organization, PROD refers to Product, and EVENT, TIME, GROUP, TITLE are code as per the self-explanatory purposes. While B-OTHER refers to any other named entities like nationalities, O refers to any special characters, emojis, hashtags and user mentions. It is worth noting that TIME excludes hours, minutes, and seconds. ‘Yesterday’, ‘tomorrow’, ‘week’ and ‘year’ are also not tagged as time. It includes only the months, days of the week, seasons, holidays and dates that happen periodically, which are not events [7].

For the current project, the test dataset was not considered, and the results and observations are solely from train and validation datasets.

DATA AUGMENTATION USING SYNONYM REPLACEMENT

We observed that the ratio of CS tweets is higher than the combined number of tweets in English and Spanish. This data imbalance could pose problems, potentially affecting model performance and leading to overfitting. To mitigate this issue, we hypothesize creating augmented data samples to equalize the number of samples across English, Spanish, and CS categories.

We use the NLTK library and Wordnet to achieve the data augmentation. We use the synonym replacement technique to synthetically generate new sentences. The core idea is to have new sentences with the same word length so that the language ID and NER tags associated with the previous word can be retained. To do so, we replace a word with its synonym which is not greater than one word. For example, the synonym of "happy" can be "full of joy". This synonym creates the imbalance between the language ID list length and the word length. It can also demand manual annotation. Thus, if "happy" is a word to be replaced, any word with the equal length is used for the substitution. Same case applies for Spanish as well.

English synonyms are picked from Wordnet and Spanish synonyms are picked from Open Multilingual Wordnet (OMW). Wordnet is an English language large lexical database. The terms in Wordnet are grouped into a set of cognitive synonyms known as synsets. Each synset is linked to synonyms, hyponyms, hypernyms, antonyms, holonyms, meronyms, entailment, cause, coordinate term and attribute [8]. Open Multilingual Wordnet is like Wordnet in the way it is structured but it encompasses large wordnets for over 26 languages and small wordnet for 57 languages [9] out of which Spanish wordnet is also a part. The synonym substitution for a sentence is done by considering all the possible sentences generated by finding all possible synonyms for all the words of a sentence. However, due to increased size of data samples, the excess sentences are ignored to maintain the equal data size that of the CS tweets.

As a result, we create a separate dataset with 8,692 data samples of each belonging to English, Spanish and CS sentences. The size of the new train dataset is now 26,076 sentences. This is used for training the neural model. Once the new dataset is created, it is saved as .arrow file in the local directory for further usage.

IMPLEMENTATION

MODEL

The neural network is implemented using the combination of XLM-RoBERTa-base as the base model and a stack of 3 Linear layers is added a custom head to the model. Each layer of Linear is separated by ReLU (Rectified Linear Unit) activation function. ReLU brings non-linearity to the network.

We chose XLM-RoBERTa-base because it is a large multilingual language model and is trained on 100 languages [10]. This feature of XLM-RoBERTa makes it a perfect candidate for our task of using CS data.

In the neural network architecture described, the first Linear layer is connected to the hidden output layers of the XLM-RoBERTa-base model, providing 768 inputs. The hidden state is configured to 256, hence the second Linear layer has 256 inputs, resulting in an output of 128. Similarly, the third layer takes 128 inputs and produces the total number of classes used for the classification task. With 19 NER (Named Entity Recognition) labels, the output of the last Linear layer is set to 19. This architecture facilitates the transformation of the input data through two linear layers, ultimately providing predictions for the classification task based on the learned features through its final linear layer.

TOKENIZATION

The dataset is tokenized using an off-the-shelf tokenizer, XLMRobertaTokenizarFast. The tokenization process must ignore the special tokens with no word IDs because we add extra padding tokens to ensure that all the samples have equal size during batching (This function is reused from Assignment 4). Special tokens have no word IDs and are assigned a label of -100. The main function is adjusted to convert label strings to their corresponding indices from the configuration's label list. During training and evaluation, tokens with label -100 are ignored for backpropagation. The same is applicable during the accuracy and Micro F1 score calculation.

TRAINING AND EVALUATION

The training and evaluation of the models was conducted on Google Colab T4 GPU configuration with free trial. To accommodate GPU computer constraints, we set a threshold improvement of 0.01 to halt the training process. This means that for training to continue, both the epoch training accuracy and the best accuracy so far must surpass this threshold value. If the model fails to meet this criterion for continuous iterations, it suggests that the model is not learning significantly to improve training accuracy, indicating the need to halt training. However, this did not affect the evaluation phase and it did for 10 epochs.

The training is done with 10 epochs. The learning rate is set to 0.0005 and the batch size is set to 256. We have used ADAM optimizer and Cross Entropy Loss as the loss function. ADAM is a gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [11]. It is also computationally and memory efficient. Cross entropy measures the difference between the predicted probability distribution of the outcome to the actual probability of true values. Cross entropy loss function ensures that the model is penalized for high losses.

As mentioned above, there are two iterations of training. One instance of a model is trained over the augmented data and the other is over the original data. The hyperparameters remain the same throughout.

After each epoch, the training loss is measured, and the predicted label and the true labels are compared to calculate the micro F1 score and accuracy. All the metrics related to the training and evaluation are discussed below.

RESULTS AND OBSERVATION

Accuracy can be defined as the total number of right predictions to the actual predictions. F1 score is the harmonic means of precision and recall. Micro F1 score, on the other hand, gives equal weightage to all the classes during the multi-class classification. It uses the net true positives, false positives, and false negatives. Net true positive refers to class-wise true positive sums calculated by merging the confusion matrix. This means it is treated one versus all corresponding to classes. As a result, NER tagging is usually evaluated using Micro F1 rather than normal F1 score. Micro F1 is defined as [12]:

$$\text{Micro F1} = \frac{\text{Net TP}}{\text{Net TP} + \frac{1}{2} * (\text{Net FP} + \text{Net FN})}$$

Augmented Dataset:

For the augmented dataset, the loss of the model for the first epoch was found to be 0.77. There was a gradual decrease in loss observed and by the end of epoch 10, the loss was noted to be 0.170.

The training accuracy was found to be 0.907 during epoch 1 and it improved after each epoch. By the end of epoch 10, it reaches 0.956. There is no threshold of 0.01 gap between the best accuracy and the individual accuracies and as a result all the epochs were trained. The Micro F1 for the training process started with 0.971 during epoch 1 and concluded with 0.975 at the epoch 10.

Now, focusing on the validation dataset, we notice the accuracy for epoch 1 to be 0.972 and ended with 0.977 during epoch 10. Micro F1 started with 0.91 in epoch 1 and ended with 0.936 for epoch 10. Each epoch took on average of 543 seconds for the execution. This includes both training and evaluation process.

Original Dataset:

For the original dataset, the loss of the model for the first epoch was found to be 0.424. There was a gradual decrease in loss observed and by the end of epoch 10, the loss was noted to be 0.086.

The training accuracy was found to be 0.963 during epoch 1 and it improved after each epoch. By the end of epoch 8, it reached 0.979. There is no threshold of 0.01 gap between the best accuracy and the individual accuracies and so the training was halted after 8 epochs. The Micro F1 for the training process started with 0.971 during epoch 1 and concluded with 0.98 at epoch 10.

Now, coming on the validation dataset, we notice the accuracy for epoch 1 to be 0.972 and ended with 0.980 during epoch 10. Micro F1 started with 0.964 in epoch 1 and ended with 0.979 for epoch 10. Each epoch took on average of 655 seconds for the execution.

The major observation that we can draw by contrasting the results from two settings is that augmented dataset didn’t perform better than the original dataset even when the data was balanced. This could be due to two reasons. One is the bigger sample size of the original dataset could have given the upper hand in the problem here. Second, the filter criteria defined by us discarded some of the influential samples for the augmented dataset and since they were available for the second setting, the model trained on the original dataset performed better in terms of both Micro F1 and accuracy. The accuracy and Micro F1 of the model trained with the original dataset indicate that the model reached the plateau of improvement at the initial stages itself but somehow had a minor addition over each iteration. All the relevant plots are attached to the project folder. These plots include the training loss, training accuracy and micro F1, validation accuracy and micro F1 for both the models. To differentiate between the model trained on augmented dataset and the original dataset, the former model is called Model 1 and the latter, Model 2.

Winata, G et al. [13] is listed in the LinCE leaderboard for the best results using CALCS 2018 Shared Task dataset for NER tagging task on XLM-RoBERTa-base model. Their F1 score on XLM-RoBERTa-base is 62.76. The difference in our performance could be due to the different methodaligies applied by them. It is also possible that the introduction of newer tokens like <USR>, <URL> and <EMOJI> has changed the outcome of the tokenization and then the model. However, this is not tried in this project and evaluating the above respects could help us understand the substantial gap in our model metrics. It should be noted that all the labels with a value of -100 were ignored during the calculation of both Micro F1 and accuracy.

Table 1 presents the contrasting results of Model 1 and Model 2 in terms of training loss, training accuracy, validation accuracy, training Micro F1 and validation Micro F1 after the end of epochs.

	Training Loss	Training Accuracy	Validation Accuracy	Training Micro F1	Validation Micro F1
Model 1	0.17	0.956	0.977	0.971	0.936
Model 2	0.086	0.979	0.98	0.98	0.979

Table 1. Performance comparison between the model trained on augmented dataset (Model 1) and model trained on original dataset (Model 2) at the end of final epoch.

DISCUSSION

It is clearly visible from the results that the proposed neural model outperformed the benchmark but there is certainly a gap in the observation due to different methodologies. It would be really advantageous to replicate the scenario proposed in [13] to better understand the emergence of this gap.

The future scope of the project is to extend the model to handle more than two languages and train it with more NER data from different languages. We can also replace XLM-RoBERTa-base with more modern language models like GPT-2 and check the performance of new system. However, it is also curious to see how the commercial versions of GPT-3.5 by OpenAI perform. To try this out, a prompt was generated with the CS sentence and the task was to identify the NER tags. This is an example of Zero-Shot prompting. The response was not accurate and vague. Next, Few-Shot prompting was tried with two sentences and highlighting which words belong to which NER category. The outcome after two-shot prompting was positive with the response being 100% correct with each word classified to its right category. These sentences were taken from the validation dataset used across the project. The result snapshot is attached along with the report. This posits a better picture of the current capability of large language models. However, it is important to introduce more new low resource languages into the setting and ensure that we have good representation of numerous languages.

REFERENCES

1. Çetinoğlu, Özlem & Schulz, Sarah & Vu, Thang. (2016). Challenges of Computational Processing of Code-Switching. 1-11. 10.18653/v1/W16-5801.
2. Winford, Donald. "Code Switching: Linguistic Aspects." An Introduction to Contact Linguistics. Malden, MA: Blackwell Pub., 2003. 126-167. Print.
3. MYERS-SCOTTON, C. (1989), Codeswitching with English: types of switching, types of communities. World Englishes, 8: 333-346. https://doi.org/10.1111/j.1467-971X.1989.tb00673.x
4. J. Li, A. Sun, J. Han and C. Li, "A Survey on Deep Learning for Named Entity Recognition," in IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 1, pp. 50-70, 1 Jan. 2022, doi: 10.1109/TKDE.2020.2981314.
5. Hugging face LINCE: https://huggingface.co/datasets/lince
6. LINCE Website: https://ritual.uh.edu/lince/datasets
7. Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, & Thamar Solorio. (2019). Named Entity Recognition on Code-Switched Data: Overview of the CALCS 2018 Shared Task.
8. Kuo-Chuan Huang, James Geller, Michael Halper, Yehoshua Perl, Junchuan Xu, Using WordNet synonym substitution to enhance UMLS source integration, Artificial Intelligence in Medicine, Volume 46, Issue 2, 2009, Pages 97-109, ISSN 0933-3657, https://doi.org/10.1016/j.artmed.2008.11.008
9. Bond, F., & Foster, R. (2013). Linking and Extending an Open Multilingual Wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1352–1362). Association for Computational Linguistics.
10. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzman, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440–8451). Association for Computational Linguistics.
11. Diederik P. Kingma, & Jimmy Ba. (2017). Adam: A Method for Stochastic Optimization.
12. Micro F1 Score: https://www.v7labs.com/blog/f1-score-guide
13. Winata, G., Cahyawijaya, S., Liu, Z., Lin, Z., Madotto, A., & Fung, P. (2021). Are Multilingual Models Effective in Code-Switching?. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching* (pp. 142–153). Association for Computational Linguistics.