spaCy Performance on Named Entity Recognition with Code-Mixed Data

BN Project Abstract for Seminar Computational Modelling (WiSe 2023/24)

Hanxin Xia | 3417418
hanxin.xia@uni-duesseldorf.de

1 Introduction

The term "named entity" was first defined in the Sixth Message Understanding Conference (MUC-6) (Grishman and Sundheim, 1996). In the publicly available description, at least three types of language expressions were categorized as such entities: "unique identifiers" of entities (organizations, persons, locations), times (dates, times), and quantities (monetary values, percentages). The extraction of such entities from a sentence is, however, not a trivial job. Outside the most obvious dictionary matching (Higashinaka et al., 2012; Shang et al., 2018), several other hybrid approaches have been proposed over the years, especially for bio-medical and other domain-specific terms (Rocktäschel et al., 2012; Lou et al., 2020). Nowadays, with pre-trained language models spaCy, StanfordNLP as such, named entity recognition tasks can be easily carried out as part of the standard pipeline during text annotation.

This research aims to investigate the multi-purposed language model spaCy's performance on named entity recognition tasks when it comes to multilingual data, above all where two languages are used interchangeably in one sentence, the so-called insertional code-switching, as follows:

(1) @esangiecarajo you asked .. *ya lo borre* so chill *jajajajaja*

The sentence starts with a matrix sentence (L1) in English. An embedded clause in Spanish is then inserted (L2, italicized). The language is switched back to English. Followed by modal particles again in Spanish. We want to know how difficult it is for a monolingual spaCy model to deal with such inserted tokens as in this sentence.

2 Research Questions

The research questions can be summarized into three major points:

Firstly, how many inserted L2 tokens that are not named entities themselves are falsely annotated as named entities by spaCy? Named entities in nature are unique to the language's vocabulary. Since the inserted L2 tokens are "foreign" to a language-specific model, we would expect that some of the unique words would be annotated as NEs.

Secondly, among the tokens that are falsely tagged as named entities, how many are just inserted non-NE L2 tokens? Namely, how many errors are caused by code-switching? By analyzing this aspect, we are able to estimate how troublesome code-switching actually is for a spaCy model. Furthermore, we would also be able to determine whether it is reasonable to use spaCy on the named entity recognition tasks of code-mixed data.

Finally, we also calculate how many inserted L2 tokens that are actually named entities are correctly identified as NEs by the L1 model. This also helps us to understand how effective spaCy is, if we are only interested in extracting NEs tokens from the embedded language.

3 Data

The data comes from Computational Approaches to Linguistic Code-Switching (CALCS) (Aguilar et al., 2018), which are accessible through the LinCE Benchmark website (https://ritual.uh.edu/lince/datasets). The specific subset used in this research is the train set in Spanish-English (SPA-ENG) from CALCS Shared Task 2018.

The original data is stored in minimal CoNLL-U format. Sentences are separated by empty lines. The tokens in the sentence are all on individual lines. The tokens are marked with language tags (`lang1` for English, `lang2` for Spanish). Whether the tokens are named entities and to what type of named entities they belong to, is also marked overtly.

Overall, there are 33,611 sentences, among which 8,692 contain both English and Spanish tokens (code-mixed). By comparing the absolute language tag amount in each sentence, we get 1,478 instances with English as the matrix language, and 7,214 are Spanish dominant.

4 Methodology

Since spaCy models are usually built on monolingual data, the choice of which language-specific model should be used to annotate the current sentence needs to be made based on individual cases. The general pipeline goes as follows: 1) Before applying the NLP model, make sure both languages are present in the current sentence; 2) Determine the matrix language (L1) of the sentence; 3) Choose the spaCy model for L1 to annotate the whole sentence, regardless of the inserted L2 tokens; 4) Retrieve named entity recognition results from linguistic features (https://spacy.io/usage/linguistic-features) built in spaCy's standard pipeline. For effeciency reasons, here we only use small variante of models for both languages: `en_core_web_sm` and `es_core_news_sm`.

Since spaCy only works on strings, the raw token chain of each sentence will be concatenated in the first step. The new text string will be passed to the language model. This, however, causes alignment issues when spaCy's tokenization results differ from the gold tokens in the original sentence dataframe. The solution is to store the named entities extracted by spaCy simply as list items. Then we go through the original token column. If the token is found in the spaCy's NE list, we mark it as `Yes`, if not, as `O`, following the original label scheme.

The focus of this experiment lies in whether the model could sufficiently distinguish between inserted code-switched normal words and named entities. The types of NEs the results tokens are assigned to are less relevant. Hence, we replace all specific named entity types in both gold tags and spaCy results with a unified value `Yes`.

For the results of each sentence, four types of information are saved: the matrix language, the gold language tags of all tokens in the sentence, the gold named entity property, and the NER results returned by spaCy model. The latter three are stored in a list aligned to the gold token list of the original sentence, on which the error analyses can be conducted.

5 Error Analyses

For the first research question, we extracted all indices of all code-switched tokens in each sentence. That is, if the matrix language of the sentence is identified as English, we extract all Spanish tokens, and vice versa. We also want to exclude cases where the inserted tokens are named entities themselves. With the list of target words' indices, the corresponding NER results from the automatic annotation will be compared. The target tokens falsely tagged as `Yes` by spaCy will be filtered out. As for results, we have 38.38% of normal L2 tokens tagged as named entities.

To investigate the relation between the performance drop and the insertions of L2 tokens, we first collect all named entity tags that are falsely given by spaCy. Then we map these errors to the language tags. If the token on which the error occurs is code-switched, it will be saved to the error list. Dividing NER errors on code-switching points by total NER errors, we get results of 27.19%. That is, over a quarter of named entity recognition errors are simply due to L2 token insertions.

Lastly, we turn to the few cases where the inserted L2 tokens are actually named entities themselves. We collect all the L2 tokens in a sentence. Then we save those that are tagged as `Yes` in the normalized gold labels. Our goal is to find out the cases where spaCy also gives `Yes` tags to these target words (L2 NEs). From the results, we see that almost 70% of these true L2 named entities are correctly tagged as NEs. The error rate is relatively low compared to earlier analyses. However, it is unclear whether this is because spaCy could overcome the "language barrier" while dealing with L2 named entities or simply by chance. Given the fact that by assigning NE tags to all indiscriminately would even achieve 100% accuracy, the actual performances of the models are still questionable.

6 Conclusion

In this experiment, we looked into monolingual spaCy language models' performances on named entity recognition tasks with code-mixed data. We see that about 40% of inserted L2 tokens are falsely recognized as named entities, which indicates that code-switching does pose a challenge to monolingual language models in NER. However, under 30% of the errors are directly caused by code-switching. To what degree CS affects performance is therefore unclear.

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, Melbourne, Australia. Association for Computational Linguistics.

Ralph Grishman and Beth M Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Ryuichiro Higashinaka, Kugatsu Sadamitsu, Kuniko Saito, Toshiro Makino, and Yoshihiro Matsuo. 2012. Creating an extended named entity dictionary from wikipedia. In *Proceedings of COLING 2012*, pages 1163–1178.

Yinxia Lou, Tao Qian, Fei Li, and Donghong Ji. 2020. A graph attention model for dictionary-guided named entity recognition. *IEEE Access*, 8:71584–71592.

Tim Rocktäschel, Michael Weidlich, and Ulf Leser. 2012. ChemSpot: a hybrid system for chemical named entity recognition. *Bioinformatics*, 28(12):1633–1640.

Jingbo Shang, Liyuan Liu, Xiang Ren, Xiaotao Gu, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary.

A Data Access

Project work and detailed instructions to retrieve data needed openly accessible under: DOI 10.17605/OSF.IO/ZSM43.