Entity named recognition summary paper

The dataset used is Cross-lingual TRansfer Evaluation of Multilingual Encoders (XTREME), This dataset consists many languages, including the four most commonly spoken languages in Switzerland: German (62.9%), French (22.9%), Italian (8.4%), and English (5.9%). Each article is annotated with LOC (location), PER (person), and ORG (organization) tags in the "inside-outside-beginning" (IOB2) format. B- prefix indicates the beginning of an entity, and consecutive tokens belonging to the same entity are given an I- prefix. An O tag indicates that the token does not belong to any entity.

-To keep track of each language, they create a Python **defaultdict** that stores the language code as the key and a PAN-X corpus of type DatasetDict as the value:

```
langs = ["de", "fr", "it", "en"]
fracs = [0.629, 0.229, 0.084, 0.059]
# Return a DatasetDict if a key doesn't exist
panx_ch = defaultdict(DatasetDict)
```

-After having tags in human-readable format in many code steps , this code will show how the tokens and tags align :

```
de_example = panx_de["train"][0]
pd.DataFrame([de_example["tokens"], de_example["ner_tags_str"]],
['Tokens', 'Tags'])
```

- Then to make sure that there is not any unusual imbalance in the tags they calculate the frequencies of each entity across each split. For the result for this step(*the distributions of the PER, LOC, and ORG frequencies are roughly the same for each split, so the validation and test sets should provide a good measure of our NER tagger's ability to generalize.*)

-Multilingual Transformers: Multilingual transformer models are usually evaluated in three different ways**. en** :Fine-tune on the English training data and then evaluate on each language's test set. **Each**: Fine-tune and evaluate on monolingual test data to measure per-language performance. **all:** Fine-tune on all the training data to evaluate on all on each language's test set. One of the first multilingual transformers was **mBERT**, which uses the same architecture and pretraining objective as **BERT** but adds Wikipedia articles from many languages to the pretraining corpus. **XLM-R** uses only MLM as a pretraining objective for 100 languages, but is distinguished by the huge size of its pretraining corpus compared to its predecessors. XLM-R is a great choice for multilingual NLU tasks.

A Closer Look at Tokenization: Instead of using a WordPiece tokenizer, XLM-R uses a tokenizer called SentencePiece that is trained on the raw text of all one hundred languages. XLM-R uses <s> and <\s> to denote the start and end of a sequence. These tokens are added in the final stage of tokenization. **The steps in the tokenization pipeline:** Normalization, Pretokenization, Tokenizer model, Postprocessing.

-After having a model and a dataset, we need to define a performance metric. **Performance Measures:** Evaluating a NER model is similar to evaluating a text classification model, and it is common to report results for precision, recall, and F_1-score. There is a nifty library called *seqeval* that is designed for these kinds of tasks.

```
from seqeval.metrics import classification_report

y_true = [["O", "O", "O", "B-MISC", "I-MISC", "I-MISC", "O"],
          ["B-PER", "I-PER", "O"]]
y_pred = [["O", "O", "B-MISC", "I-MISC", "I-MISC", "I-MISC", "O"],
          ["B-PER", "I-PER", "O"]]
```

-For a result of this code : *seqeval* expects the predictions and labels as lists of lists, with each list corresponding to a single example in our validation or test sets. To integrate these metrics during training, they use a function that can take the outputs of the model and convert them into the lists that *seqeval* expects. Equipped with a performance metric, The next step is actually training the model.

-The first strategy will be to fine-tune the base model on the German subset of PAN-X and then evaluate its zero-shot cross-lingual performance on French, Italian, and English. they use the Transformers Trainer to handle training loop, first they define the training attributes using the TrainingArguments class.

-Error Analysis: Examples where training can fail include: We might accidentally mask too many tokens and also mask some of our labels to get a really promising loss drop.The compute_metrics() function might have a bug that overestimates the true performance.We might include the zero class or O entity in NER as a normal class, which will heavily skew the accuracy and F1-score since it is the majority class by a large margin.