PPD French Named Entity Recognition with Transformers architectures

Université de Paris
MLDS master (Machine Learning for Data Science)

Laura Liepchitz
laura.liepchitz@gmail.com

Amar Litim
ammar.litim@gmail.com

Maxence Thiry
maxence.thiry@gmail.com

Supervisor: François Role, Ph.D
francois.role@parisdescartes.fr

May 24, 2020

Abstract

Natural Language Processing (NLP) is booming as the use of text data is becoming more and more common. Analyzing a text is a complicated task that requires a lot of research beforehand. Many research documents exist on English, which is not yet the case for French, a very complicated language. However, over the last two years, teams from INRIA, for example, have been working on this task by creating models based on the BERT architecture. These models are very recent and there is little feedback on them. We have therefore tested different functionalities in order to measure their performance.

1 Context and Introduction

In the context of the Master's Degree on Machine Learning for Data Science (MLDS), at the University of Paris (formerly, Descartes)[1], we have been given the task to discuss about the latest advances in nlp.Indeed French language models have recently been published. CamemBERT and FlauBERT share the same observation: most of the models were formed either on English data or on the concatenation of data from several languages, which limits their use for other languages. To try to solve this problem, researchers have worked with Transformer models to perform tasks such as part marking, dependency analysis, named entity recognition and natural language inference. Transformer is an architecture for transforming one sequence into another using two parts (encoder and decoder), but they differ from sequence-to-sequence models because they do not involve a recurrent network. CamemBERT is trained on the OSCAR corpus, which is the French part of the CommonCrawl dataset. Indeed, Camembert "use a non-shuffled version of the French data, which amounts to 138GB of raw text and 32.7B tokens after subword tokenization"[1]. In the CamemBERT paper, their models outperform the existing models. This work on French language is very positive because even advanced tools such as Spacy are still poorly adapted to deal with languages other than English.

[1] MLDS

These models are relatively new and still poorly documented. Our text mining teacher, François Role, asked us to test the different functionalities of the models, to measure their performance and to compare them with each other.
We will then focus on entity detection (NER for Name Entity Recognition). This function is used to detect entities, a kind of keywords in documents. These entities are labeled and then added via Transformer to a model such as Camembert to learn and test entity detection. Entity detection can easily be tested on the presentation page of the API Natural Language from Google. By giving a sentence, it will detect entities. By using the Europeana dataset we will be able to test the NER on different models and of course, compare their performance.

2 Methodology and Metrics Overview

2.1 Tools and Concept

Transformers and Simple Transformers
Before talking about transformers, we have to talk about sequence to sequence algorithms. Sequence-to-sequence (Seq2seq) is a neural network who transforms a sequence of elements into another sequence. Seq2Seq models are based on an Encoder and a Decoder.The Encoder will take the input sequence to map it into a higher dimensional-space. In the other hand, the Decoder is feed from the Encoder which turns into an output sequence into the form needed (translate for example). Usually behind the seq2seq there is a LSTM neural network.
Another point of interet is the "Attention". Attention will determine which part of the sequence is important. The encoder will identify key words that are important for the semantics of the sentence, and gives them to the decoder in addition to the normal translation. With this important key-word detection, the decoder will know which parts of the sentences are importants
The concept of transformers has been imagined in the document *Attention is all we need* where the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely[2]. The transformers were therefore designed mainly to process sequences and in particular to translate text. One of the first applications was BERT[2]. To resume Transformers provides state-of-the-art general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, T5, CTRL...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over thousands of pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch[3].
To use transformers we can use pipelines. This is a simple way to call up the models, see tab. 1. It will reduce the code of the models to provide a much simpler API to use. It will simply run a defined model. It is possible to test transformers on this site
We've been trying to launch several models with several pipelines :

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Feature	Return	Function	Model	Input
Feature extraction	StartFragmentFeatureExtractionPipeline	Extracts the hidden states of the base transformer, which can be used as functionalities in downstream tasks.	Model	Sentence
Sentiment analysis	TextClassificationPipeline	Classify sequences according to positive or negative feelings	ModelForSequenceClassification	Sentence
NER	NerPipeline	Named Entity Recognition: predicts the classes of certain words in a sentence: person, organization, location or miscellaneous	ModelForTokenClassification	Sentence
Question answering	QuestionAnsweringPipeline	Answers questions in a specific context	ModelForQuestionAnswering	Question & context
Fill-mask	FillMaskPipeline	Predicts the masked words of a sentence.	ModelWithLMHead	Sentence
Summarization	SummarizationPipeline	Summarizes press articles and other documents	Model ?	Sentence
Translation xx to yy	TransformationPipeline	Translates from one language to another	Model ?	Sentence and translation language

Table 1: Feature and model per pipeline

The simple Transformers library is use to make Transformer models easiest to use. Indeed, only three lines of code are now enough to launch a Transformer model. The simple problem with simple transformers is the lack of certain Transformer features. Not all functions are available yet, but NER is supported.

NER

In the broadest sense, entity detection allows an automated and intelligent analysis of a text, which is especially useful for long and complex documents. "The goal is to locate and classify named entities in a sequence. The named entities are pre-defined categories chosen according to the use case such as names of people, organizations, places, codes, time notations, monetary values, etc. Essentially, NER aims to assign a class to each token (usually a single word) in a sequence. Because of this, NER is also referred to as token classification."[4]

In NLP, Named Entity Recognition is an important method in order to extract relevant information. For general entity such as name, location and organization we can easily use pre-trained model but for specific domain entity, we have to labeling some entities so that we can recognize those entity. For English NER, there is a lot of package with pre-train model like in spaCy package[3] or Stanford NER package[4] but it's very difficult for french.

[3] spaCy
[4] Stanford NER

3 Dataset

The Europeana Newspapers French text are extracted from the BnF newspapers collections processed during the Europeana Newspapers project. This dataset contains 212 annotated files of 1,000 words each. They are extracted from the Gallica's digitized press corpora processed during the Europeana Newspapers project between 1870 and 1945. The data set consists of text annotated with 3 terms of named entities following the IOB format :

- Person as "PER"
- Location as"LOC"
- Organisation as "ORG"

The tags have a prefix of either B- for beginning of named entity, I- for inside of named entity. The label O for outside of named entity tags are used to mark tokens that are not a named entity.
Below as tab. 2 is an extracts from the dataset and the figure is the distribution of the different entities (fig. 1). We can see that the majority of the names are not labeled

Word	Entities
comme	o
il	o
le	o
déclarait	o
à	o
Paris	I-Lieu
Michel	I-Pers
est	0
bien	0

Table 2: Europeana Newspapers French extract

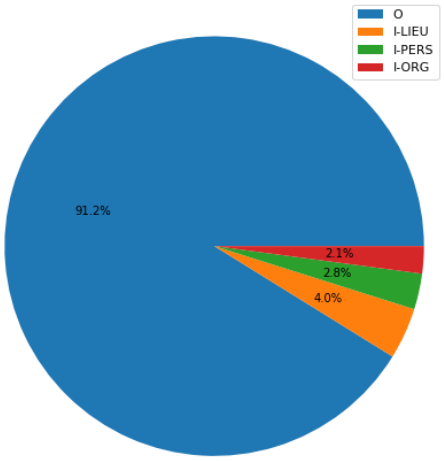

Figure 1: Distribution of the different entities present in the Europeana Newspapers French data sets

3.1 Methodology

The Europeana Newspapers French being composed of 212 files, the first step is to concatenate them to create the database for the experiment. Some data processsing was necessary. We started by checking that each word in the Europeana Newspaper French has only one label associated with it. Different labels were associated with the "!" symbol ("O", "I-PERS", "I-LIEU"). We replaced them everywhere by the tag "O" only. In order to be able

to test the models afterwards, the dataset had to be put in the right format. That is to say with the columns "sentence_id", "words" and "labels". The column "sentence_id" did not exist in the dataset, so it had to be created. The "sentence_id" is the sentence's number to which a word belongs. A sentence is considered to begin when one of the following separators is encountered : ".", "?" and "!".
Once the clean database is obtained, different NER models can be tested and compared with each other (Table 3).

Family Model	**Model code for NERModel**
BERT	bert
CamemBERT	camembert
RoBERTa	roberta
DistilBERT	distilbert
ELECTRA	electra
XLM-RoBERTa	xlmroberta

Table 3: NER models available in Simple Transformer

Several functions have been implemented to facilitate and accelerate the testing of these models. They are described in the "Functions" section of the project's notebook [5].
For each family of models, the basic model is tested. To go further we tested other pre-trained models of the most performant family. All of the models are all available on the Hugging Face platform [6].
In order to detail the overall performance of the models we examined at the performances by entity.
The results are detailed and discussed in the dedicated sections.

3.2 Metrics

Seqeval is a Python framework for sequence labeling evaluation. It can evaluate the performance of chunking tasks such as named-entity recognition, part-of-speech tagging, semantic role labeling and so on.
Multiple tagging format are supported such as IOB1,IOB2,IOE1,IOE2 and IOBES.
Instead of evaluating predictions at token-level (like in the scikit-learn library), seqeval allows to evaluate predictions at entity-level. For example, the entity Person will be evaluated as one class gathering B-PER and I-PER. The following metrics are supported by seqeval to evaluate models'performance and are used in our work to compare models with each other.

Accuracy

The accuracy is the ratio of the number of correct predictions to the total number of predictions : $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

[5] Project notebook
[6] Hugging Face models

Precision

The precision is the number of correct positive results divided by the number of positive results predicted by the classifier : $Precision = \frac{TP}{TP+FP}$

Recall

The recall is the number of correct positive results divided by the number of all relevant samples (real positive samples) : $Recall = \frac{TP}{TP+FN}$

F1-score

The F1-score is the harmonic mean between precision and recall. It is calculated as follow: $f1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

Classification Report

The Classification Report uses the previous metrics and calculates them by entity.

4 Results

We first studied all the base models available for the NER function. As a reminder, performance is measured using Eval_loss, F1_score, Precision and Recall. The following table 4 summarizes the performances :

Eval_loss	F1-score	Precision	Recall	Model Type	Model Name
0.2	0.67	0.67	0.67	bert	bert-base-cased
0.2	0.72	0.73	0.72	camembert	camembert-base
0.23	0.6	0.6	0.6	roberta	roberta-base
0.21	0.62	0.63	0.61	distilbert	distilbert-base-cased
0.27	0.39	0.44	0.35	electra	google/electra-base-discriminator
0.2	0.7	0.68	0.71	xlmroberta	xlm-roberta-base

Table 4: Performances of the base NER models available in Hugging Face

Globally, the Camembert model is the one performing best, all metrics combined. So in a second step, we focused on all the NER Camembert models available on Hugging face.
Today there are a large number of Camembert models. They are all available on Hugging Face. Camembert provide a model to train to achieve what you want. It is up to the user to use the model and train it as he wants. It is possible to choose all the parameters of the model and the training games to train the model. The pre-provided models have been created by the community or Camembert researchers.
The difference between them is the result of different training (different data sets, parameters, etc.). Their performances are summarized in the table 5:

Eval_loss	F1-score	Precision	Recall	Model Type	Model Name
0.23	0.6	0.58	0.62	camembert	idb-ita-gilberto-uncased-from-camembert
0.18	0.76	0.74	0.77	camembert	camembert-camembert-large
0.19	0.73	0.73	0.72	camembert	fmikaelian-camembert-base-fquad
0.21	0.71	0.71	0.71	camembert	illuin-camembert-large-fquad
0.2	0.71	0.72	0.71	camembert	fmikaelian-camembert-base-squad
0.21	0.67	0.7	0.64	camembert	illuin-camembert-base-fquad
0.21	0.67	0.7	0.64	camembert	camembert-camembert-base
0.19	0.73	0.74	0.73	camembert	camembert-camembert-base-ccnet
0.19	0.74	0.75	0.73	camembert	camembert-camembert-base-oscar-4gb
0.19	0.72	0.73	0.72	camembert	camembert-camembert-base-ccnet-4gb
0.2	0.71	0.72	0.7	camembert	camembert-camembert-base-wikipedia-4gb

Table 5: Performances of the Camembert NER models available in Hugging Face

The model camembert-camembert-large returns the best results all metrics combined with a F1-score at **76%** which is an improvement compared to Camembert-base whose F1-score was **72%**. To better understand the overall performances and to detail them, we examine the performances by entity. The following figure (fig. 2) is obtained by using the classification report function from seqeval package with the model camembert-camembert-large :

```
F1-score :  0.029169181251257288
Accuracy :  0.847561606422379
Classification Report :
              precision    recall  f1-score   support

        LIEU       0.03      0.03      0.03      2095
         ORG       0.02      0.02      0.02       738
        PERS       0.03      0.03      0.03      2093

   micro avg       0.03      0.03      0.03      4926
   macro avg       0.03      0.03      0.03      4926
```

Figure 2: Classification Report

The tabular (tab. 2) shows that performance by entity is very poor. The only acceptable value is the accuracy but accuracy can be a misleading metric for imbalanced data sets. In this case, it takes into account the words without entities (labeled "O") which are by far the majority.

The global f1-score calculated via the seqeval package is much lower than the one calculated directly by the eval_model function of the simpletransformers library (**3%** vs. **76%** respectively). The calculation of Simple Transformers metrics is however supposed to be based on seqeval. We therefore do not understand where these considerable differences in results come from.

5 Discussion and Conclusion

Camembert-large is the best performing model for NER. However, we get lower performance than the one described in the paper[1]. They have a F1-score of 89% whereas we only have 76%. Thanks to simple transformers, it is relatively easy to use. However we have noticed some topics of discussion :

First, the training dataset quality. The learning dataset must have sentences composed of word sequences. These words are labelled as necessary. The first problem encountered is the lack of labelled words. In our sample, only 10% of the words are labeled. This has a big impact on the training and test dataset. Also, we use 80% of the sequence_id for training and 20% for testing but we don't take into account the fact that some sequences don't have the same number of words. To help, methods for re-balancing the training data set should be tested, for example with a WHILE loop, as long as a proportion of about 80% of words in the training set is not reached. We can also work with undersampling methods[5]. There's also a problem with punctuation. We have cut out the sentences with the punctuation ".", "!", "?" to create id sentences. However, we noticed that there were often punctuation marks in the middle of the sentences. This must have an impact on learning because it breaks the syntax of the sentence. For example: "Since the last night, Mr. John has had a sore neck". Punctuation after the "M" breaks the sentence in half. A way should be found to solve this problem to ensure that each sentence remains whole.

Second, the vocabulary and entity diversity. As explained above, the learning dataset is a sample of French articles on various common topics. The model will learn from this information. But it will learn about very generic topics. This is one of the first limitations of using this type of method. The vocabulary is too generic to adapt to a specific subject such as a business problem. You should also add a lot of categories in order to use it to the maximum.

Third, the uppercase letters. In our database there are 3 categories of labels (person, location and organisation). After some tests we realized that it was quite simple to tromble the label of the model only by removing the capital letter at the word. **Forth, the code**. We encountered problems installing the packages which forced us to work on Google Colab. In order to test the performance of all NER models, we set up a kind of grid search. However, all the tensors are stored in the cache of the graphics card, which saturates it after a while. On Colab, graphics cards are hardware that is shared between several machines and therefore several users, so it is not possible to force the cache to be emptied.

Fifth, the metrics by entities. We have tried several metrics to measure model performance. In a second step, we wanted to see the performance by entities and more globally. For that we calculated the different values necessary as described in the following description in 4. By studying the metrics we could see that the performances were very bad according to the "classification_report" function. Due to lack of time, we could not understand where the performance problem came from and why the function returned these values.

Sixth, other Bert model. We looked to apply NER with Flaubert's models. However it is

not supported at the moment on Simple Transformers. It is only supported for Text Classification. Moreover we notice that Hugging Face does not provide for Flaubert Token-Classification models. We found a paper that presents both models. According to it, "the performance of FlauBERT is overall better than camemBERT. [...] Their results show again that a French language model improves the results compared to similar BERT (multilingual) models as well as to other French-based models. The performances of FlauBERT and CamemBERT are very close. Unfortunately, FLUE does not include a NER benchmark in FLUE. All in all, FlauBERT performs very similarly to CamemBERT while it is trained on fewer data."[6] However, they have not carried out a benchmark on NER.
NERs are a very fast method to detect keywords in corpus. These keywords are fundamental in sentence completion exercises but also in translation and syntax study. The models based on Bert and obviously Camembert are a great step forward in the treatment of French linguistics. This field is still too little treated. We were very happy to work on this package to test its performance and its uses. We had problems using the functions and choosing the best parameters with the basic library but simple transformers greatly simplify the work. The tests on the NER are very good. In a professional context, some functions would be very useful like sentence completion. Entity detection on the other hand is too generic to be used as such. A large dictionary specialized in the field of study should be created and labeled beforehand, it would take a lot of time. It may still be too early to use Camembert in a professional environment. Its use is still too limited but we are convinced that the package will quickly become very versatile.

If we had more time on this project we would have liked to understand why the *cambert-camembert-large* model performs better than other Camembert models. More precisely why the large architecture, the training on the CCNet dataset and a model with 335 million parameters allows us to get better performance on our dataset. Moreover, in a professional context, it would be interesting to study the computation time of each model. These benchmarks can be interesting when a professional context where it is often asked to have fast results more than the best results.

Bibliographie

[1] Louis Martin et al. "CamemBERT: a Tasty French Language Model". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.

[2] Ashish Vaswani et al. *Attention Is All You Need*. 2017. eprint: arXiv:1706.03762.

[3] Rmroczkowski. *rmroczkowski/transformers*. 2020. URL: https://github.com/rmroczkowski/transformers.

[4] Thilina Rajapakse. *Named Entitty Recognition Specifics*. 2020. URL: https://simpletransformers.ai/docs/ner-specifics/.

[5] Abbas Akkasi. *Sentence-based undersampling for named entity recognition using genetic algorithm*. Mar. 2018. DOI: 10.1007/s42044-018-0014-5.

[6] *French language keeping pace with AI: FlauBERT, CamemBERT, PIAF*. URL: https://piaf.etalab.studio/francophonie-ia-english/.