Text classification tutorial
Machine Learning for Natural Language Processing 2021

Daphné Le Cornec
ENSAE
`daphne.lecornec@ensae.fr`

Abstract

Text classification is an important task in Natural Language Processing (NLP). Its main applications are sentiment analysis (which can be used, for example, for website moderation or marketing), intent detection or document clustering (such as mail/spam classification). For this reason, text classification is a good way to get started in NLP. The Pytorch library proposes a tutorial (Pytorch) with this in mind, yet has a limited scope and some weaknesses regarding the data acquisition, the model used or the lack of hindsight on the results. In this work, we present a revised version of the Pytorch tutorial which covers all building blocks of a standard NLP project.

1 Introduction

NLP is a specific field of Machine Learning which consists of several tasks, including text classification. To get started on this specific task, Pytorch proposes a name classification tutorial that shows how to build an NLP pipeline. However, this tutorial lacks important features, namely: 1. data acquisition: in today's world, most NLP work starts with gathering data; 2. training pipeline: a robust project rely on a number of well defined pipelines such as dataloading, model training, evaluation; 3. models: Pytorch tutorial does not introduce SOTA models (LSTM, Attention) but only uses an RNN; 4. results: finally, NLP work does not consists only in training a model but also in interpreting results. In this project, we propose an updated version of this tutorial [1] by including all main features described above.

2 Data acquisition

Our goal is to classify names by country of origin. To this end, we gather a list of names and their associated country from the following website `https://adoption.com/baby-names/` using packages *request* and *Beautiful Soup*. The package *request* is used to query webpages, whereas the *Beautiful Soup* library contains tools to parse and scrap webpages. Each name is linked to a meaning, a gender and an origin. In our classification task, only the name and its origin are of interest. Data has been processed to create a clean and easy to deal with database, linking names to their origin. It should be noted that a name can be associated with various origins.

3 Training Pipeline

The proposed framework reflects the building blocks of a standard NLP pipeline: 1. vocabulary, dataset and dataloader; 2. models and trainer; 3. evaluation procedure. The `Vocabulary` is constructed from the inputs' characters since the tokenization is done at the character level. This `Vocabulary` enables the creation of a smooth dataloading pipleline including a `Dataset` which can serve training examples one by one and a `Dataloader`. The `Dataloader` handles examples by batch, using padding as necessary via a custom collate function (special padding characters are added to shorter names so all input tensors have the same length). Once the embeddings and models are instanciated, they are trained using the `Trainer` pipeline. The latter supports early stopping in order to avoid overfitting: during the training phase, the model is evaluated on a held-out set; the training stops either when the number of epochs is reached or when the performance of the model (based on the evaluation score) does not improve for a certain period of time (called *patience*). This framework ends with the `Evaluation` process including quantitative and qualitative analy-

[1] `https://github.com/DaphLC/name-classification`

sis. These pipelines define the complete framework allowing to perform the classification task.

4 Models

In this project, we explore two embedding strategies and two neural architectures.

Embedding strategies. Two embedding models are implemented: an embedding matrix learned during the task (using *torch.nn.Embedding*) and an embedding following the Word2Vec method (Mikolov et al., 2013) but trained at the character level (hence, called *Char2Vec*). The Word2Vec algorithm produces word embedding based on contextual information. In our *Char2Vec* model, we use a context window of 3 characters: the context used to learn the embedding of a specific character are the characters on its left and right. It seems reasonable to use a small window for our task as names are not usually long words.

Neural architectures. For our classification task, we focus on two types of models, namely a bidirectional Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) with attention mechanism (Bahdanau et al., 2016) and a Multi-head Self-Attention model (Vaswani et al., 2017). An LSTM model is a Recurrent Neural Network able to learn long-term dependencies. It is said to be bidirectional when it runs through inputs in both directions (from left to right and from right to left). The Self-Attention network is based on the attention mechanism, each token of the input is composed of a Key, a Query and a Value through which it can interact with other tokens of the input. We include positional encoding in our model which enables to take into account the place of tokens in the input. Both models are trained using the Adam optimizer and Cross Entropy Loss.

5 Results

Using the above framework, a large range of experiments could be performed. Here, we focus on comparing embedding strategies and neural architectures across a single task. We sample 2000 names from four fairly distinct origins: French, English, Arabic and Greek, and train the models to recover the origins. The evaluated models are an LSTM and a Self-Attention model with pretrained embeddings, and, an LSTM and a Self-Attention model with Char2Vec embeddings. Classes are balanced in the experiments. The fact that multiple targets exist for one name is taken into account in the evaluation process: a prediction is considered to be correct if it is included in the list of admissible targets for the input.

Model	Embedding	Accuracy score
LSTM	Pretrained	**78.92%**
LSTM	Char2Vec	76.50%
Self-Attention	Pretrained	65.79%
Self-Attention	Char2Vec	67.96%

Table 1: Accuracy scores allowing multiple outputs

5.1 Quantitative Evaluation

The quantitative analysis of the experiments is performed using accuracy score, which is a common metric of classification tasks. The scores of each model are given in Table 1. We can assess the performance of a classification model with a confusion matrix as it provides a detailed analysis of prediction results by class (Appendix A).

5.2 Qualitative Analysis

A standard method for qualitative analysis is to use the attention weights (when provided by the model) to extract information on the tokens on which the model relies the most to make its prediction. Since both models, LSTM and Self-Attention, include a layer of attention, we can visualize which tokens are considered important when predicting the origin of a name. An example of visualization is provided in Appendix B. As can be seen, the LSTM model seems to use one token to make its prediction, whereas the Self-Attention model seems to rely on several characters.

6 Discussion/Conclusion

In conclusion, this tutorial is a great introduction to NLP as it enables to perform a text classification project entirely by covering all features of NLP: tokenization, embeddings, NLP models, attention, etc. The goal here was not to have a high-performance model, but to explore in greater depth the techniques introduced in class.
An enhancement of the proposed implementation would be to take into account in the training phase the possibility of having multiple targets. Moreover, an interesting addition to this tutorial would be to include a text generation task: the goal would be to generate a name from an origin.

References

Pytorch. Nlp from scratch: Classifying names with a character-level rnn.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

A Confusion matrices

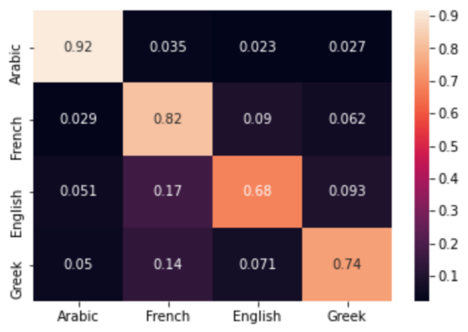

Figure 1: LSTM with pretrained embeddings

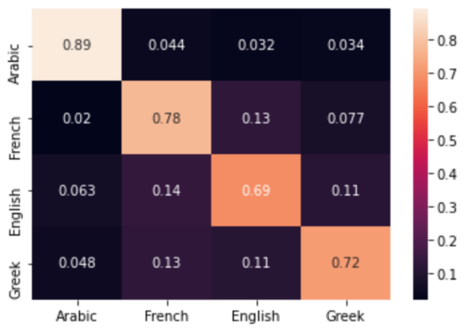

Figure 2: LSTM with Char2Vec embeddings

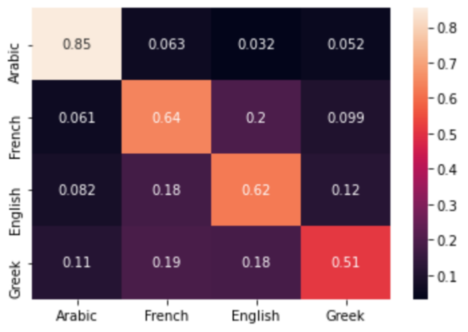

Figure 3: Self-Attention with pretrained embeddings

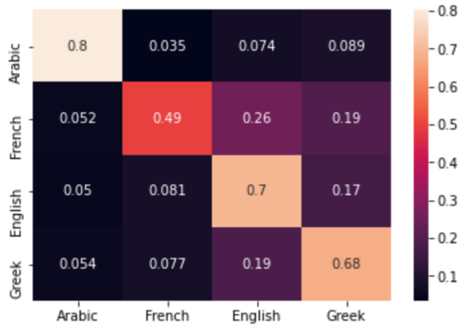

Figure 4: Self-Attention with Char2Vec embeddings

B Interpretation via visualization

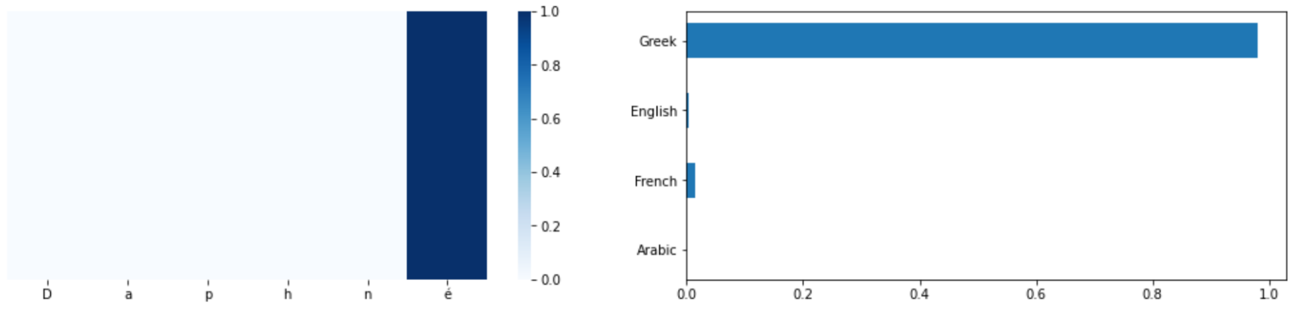

Figure 5: LSTM with pretrained embeddings

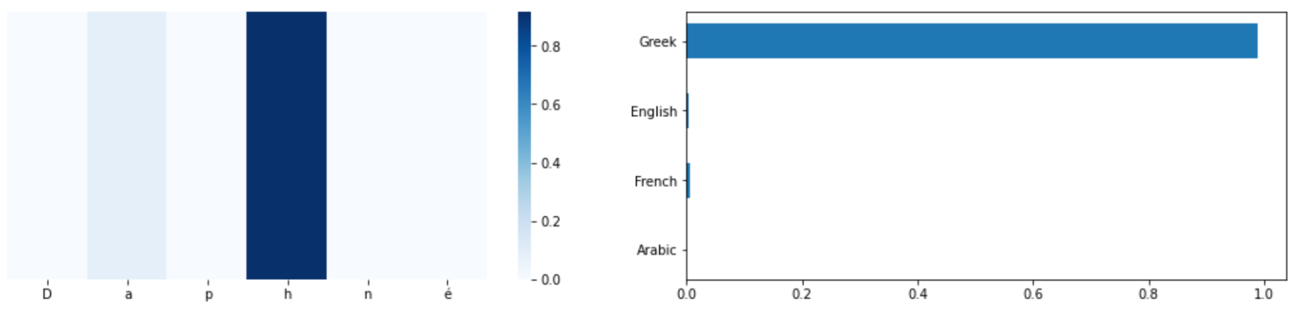

Figure 6: LSTM with Char2Vec embeddings

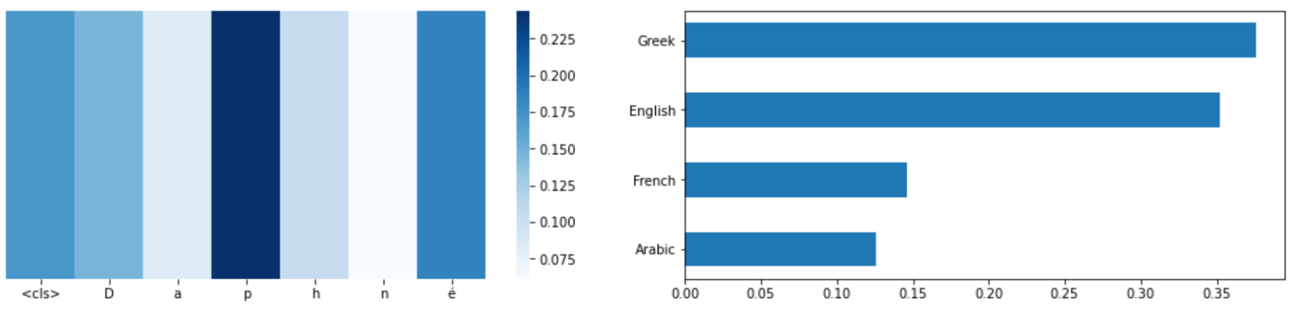

Figure 7: Self-Attention with pretrained embeddings

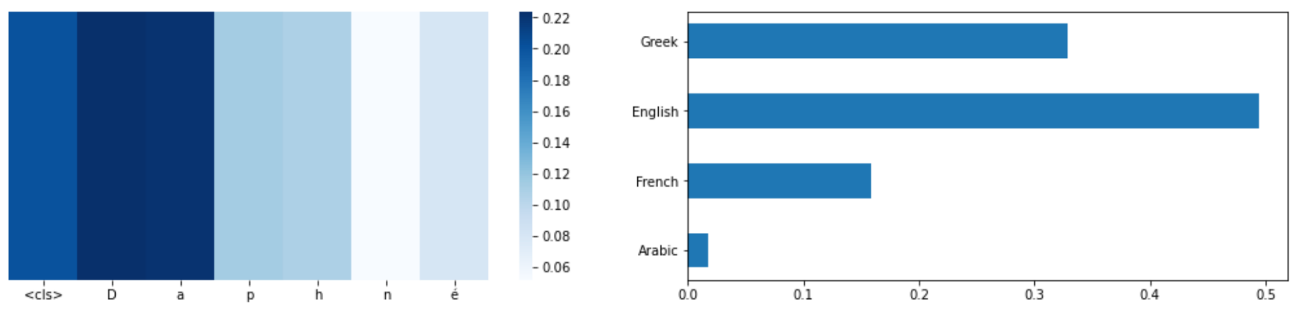

Figure 8: Self-Attention with Char2Vec embeddings