Job Title Classification
Machine Learning for Natural Language Processing 2021

Alexandre Dupuy-Zini
ENSAE Paris
alexandre.dupuy-zini@ensae.fr

Ridouane Ghermi
ENSAE Paris
ridouane.ghermi@ensae.fr

Abstract

This project aims at evaluating models of text representation and classification algorithms to match a job title to its more general category. We use an official job hierarchy defined by the *Pole Emploi*, and the performances are evaluated in a quantitative and a qualitative ways. Simpler models like a word frequency based one are compared to state-of-the-art models like BERT in order to see their difference of application. Finally, the best model is applied to a classification task with more specific and numberous classes.

1 Introduction

Grouping jobs by theme is important to bring job offers and candidates together. However, everyone is free to use the terms they wish to describe a job. The *ROME (Répertoire Opérationnel des Métiers et Emplois)* job inventory is therefore built as a tool to formalize the designation of skills. We wish to know if it is possible to predict the class of a job from its title, or even to predict a classification for an alternative title or for a new job.

We will use data from the *ROME* inventory available and explained on the Pole Emploi website[1].

We will first explore the whole inventory to frame the task, and then build different classifiers from the simplest to the more advanced. As there are several levels of more or less generic job categories, we will observe the performances on these different levels. Every model will be evaluated quantitatively and qualitatively. Finally, we will draw conclusions from our work and discuss possible extensions.

A notebook containing all the mentionned code and results is available on Colab[2] and Github [3].

2 Problem Framing

The *ROME* inventory is a tree built with several levels of incremental specificity. It classifies 11112 job titles in the following hierarchy :

Level	Number of branches	Name
1	14	Large Domains
2	110	Professional Domains
3	530	ROME Codes

The majority of job titles are both masculine and feminine with the two writings separated by a slash. Moreover, the labels are unbalanced for all the levels of this tree, and the dataset contains only 7294 unique words. Therefore, the corpus is small and the job titles are mostly very short with less than 10 words in general. The following table gives an example of a random job title with its ROME code.

Rome Code	I1401
ROME Code Name	Maintenance informatique et bureautique
Job Title	Agent / Agente de maintenance sur systèmes d'impression et de reprographie

First, we evaluate a large set of models of text representation and classification algorithms to match a job title to its large domain class. Then, the best model can be evaluated on a larger number of more specified classes.

3 Experiments Protocol

The task is a multiclass classification problem with a potentially high number of classes. However, there is a need for a clear and easy point

[1] https://www.pole-emploi.org/opendata/repertoire-operationnel-des-meti.html?type=article

[2] https://colab.research.google.com/drive/1ynpMShHuNGJeSbsO4eyEBzxxhmyQchZp?usp=sharing

[3] https://github.com/alexandredupuy-zini/ML_for_NLP

of comparison between all the models used. We will here use the f1-score, with different weighting strategies, and the Multiclass LogLoss. Multiclass LogLoss is usefull when all the prediction probabilities are important, whereas the f1-score will only take into account the highest probability class. Another simple metric used is the accuracy on a validation set.

Still, one can qualitatively assess the performance of the models by manually testing some out-of-dataset job titles, using for example synonyms. Here, we use 4 invented examples given in Appendix 1.

Before building any model, all job titles are quickly preprocessed with care not to remove too many words in the sequence since they are already short. The dataset is then split into test and train subsets, with a total of 6033 unique words (fig. 1).

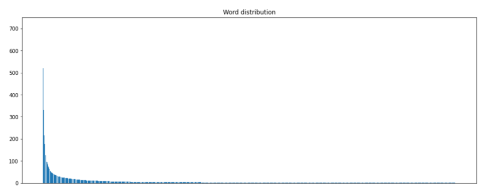

Figure 1: Word distribution after pre-processing.

The protocol therefore consists in testing the simplest models to the most sophisticated on the classification in one of the 14 large domains. The simplest one is to directly use the word distribution of each class by comparing the word frequencies. This helps achieve a baseline score. Then, classification models are trained on different types of embeddings : Bag-of-Words (BOW), tf-idf, and Word2Vec (Mikolov et al., 2013). In particular, we compare the performances of our own embeddings trained on the corpus with pretrained embeddings. As the embeddings are at the word level, and not at the document level, we use the average of the words to create a sequence embedding. Then, we use Doc2Vec (Le and Mikolov, 2014) that provides document embeddings by following a similar process as Word2Vec does at the word level. Finally, we try neural networks on that task. The first neural network is a Seq2Seq model using LSTM, and the second one is a Sequence Classification model based on a pretrained BERT (Devlin et al., 2019) model called CamemBERT (Martin et al., 2020). However, as we don't focus on finding the best hyperparameters for each model, all performances could be improved.

In addition, the best model is trained on a classification task with 110 classes to test its scalability.

4 Results

A table of the best models per type of embedding/model is available in Appendix 2. Most models provide good performances quantitavely, with the BERT model being the better one. In general, we see that using pretrained embeddings help achieve better results than homemade embeddings since the vocabulary and corpus is limited. In particular, we tried to see if the embeddings could be separated enough for an easy classification model by using a t-SNE, but that never gave any good visual.

Moreover, not all models are flexible enough to be adapted to new entries. In general, the models using pretrained embeddings achieved the best scores, but only one model got a 100% success rate. This could be an overfitting problem in some cases, like for the BERT model.

Finally, coding all those models helped realise that some models need more preprocessing that others. For BOW or tf-idf, the size increases greatly when the vocabulary increases, but Seq2Seq or BERT are less sensitive to that parameter, and can digest longer sentences with a richer vocabulary.

The BERT model also happened to be a great classifier for the larger classification task, by showing inferior but similar scores than on the previous task. As expected, convergence is slower than for less classes.

5 Discussion/Conclusion

As a conclusion, some models performed poorly but the more sophisticated ones performed well in most cases. In particular, BERT and a SVC on Word2Vec pretrained embeddings achieved great scores. Training our own embeddings was generally associated with a lower performance, while transfer learning help bring context and knowledge to a small corpus and vocabulary.

Finally, this work could be carried out in order to fastly classify new job titles without necessarily asking a panel of experts. As an extension, the best models could be used to automatically classify the job offers in defined labels that candidates research.

6 Appendix

6.1 Qualitative testing examples

Job Title	Large Domain Class	Idea
présentateur journal télévisé	Communication, Média et Multimédia	Close rewriting of a dataset item
présentateur bulletin météo	Communication, Média et Multimédia	"météo" appeals to other large domains than media
animateur centre loisirs pour enfant	Hôtellerie-Restauration, Tourisme, Loisirs et Animation	Out-of-dataset job title
responsable garde accueil gestion immeuble	Services à la personne et à la collectivité	Definition of "concierge"

6.2 Metrics associated to the best models

	word frequency	BOW+LogReg	tfidf+SVC	pretrained+SVC	trained+RFC	LSTM	BERT
micro f1		0.781	0.757	0.799	0.511	0.779	**0.853**
macro f1		0.764	0.741	0.788	0.439	0.744	**0.837**
weighted f1	0.726	0.783	0.754	0.798	0.498	0.779	**0.853**
logloss		0.801	0.704	**0.700**	2.3		
accuracy	0.72	0.781	0.757	0.799	0.511	0.779	**0.853**
qualitative	0/4	0/4	1/4	**4/4**	1/4	1/4	2/4

References

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

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. Camembert: a tasty french language model.