POSOS challenge — Report

Hugo Cisneros

March 2019

Abstract

This paper describes a data anlysis and model selection process carried out as part of the *Challenge Data* of Stéphane Mallat's course *Apprentissage par réseaux de neurones profonds*. The goal of the POSOS challenge is to classify and predict the intent of drug related questions from a dataset provided by the company POSOS. The best score was achieved with a Logistic regression applied to a tf-idf representation of the pre-processed sentences.

1 Data analysis

The base dataset is composed of 8028 training sentences and 2035 test sentences. I further randomly split the training set into a validation set of size 1606 and a new training set of size 6422 for development purposes. Models should be trained on the smaller training set and selected with the "unseen" validation test to ensure that no overfitting has occurred during training. There is always a risk of indirectly overfitting the validation set itself by iteratively selecting the models that gives the best results on it, and a more rigorous method is to use multiple cross-validation splits and compute the averaged score of the models on these splits.

1.1 Overview

The main characteristic of this dataset is its very high number of classes compared to the total number of elements in the set. One of the main consequence of this repartition is the presence of classes with very few examples which can lead to highly biased estimations of the performance of classification algorithms.

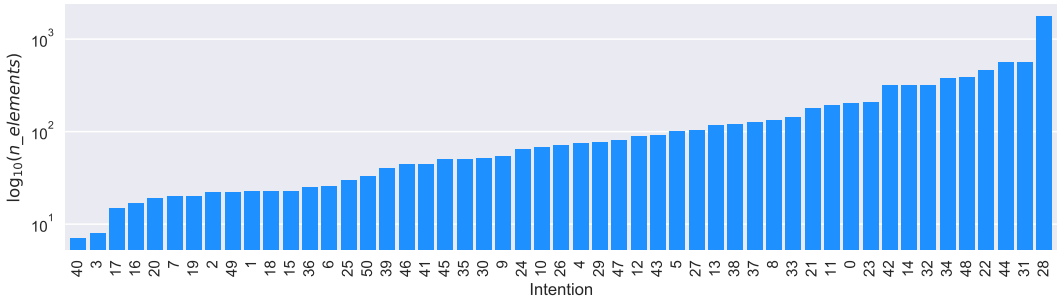


Figure 1: Sorted number of elements per class in the training set (in log-scale)

There is a great disparity in the number of elements per class. Figure 1 shows the sorted number of elements per class in log-scale, highlighting a more than 100 fold difference in class

size between the most and least populous classes. This fact is a challenge in itself, because most models struggle to get satisfying predictions from very unbalanced classes.

The classifier will need to deal with complex decision functions, as simply working with the presence/absence of words and expressions probably won't yield satisfying results, especially for very small classes where there is not enough content to base an algorithm on unigrams/bigrams.

1.2 Text pre-processing

Input text from the dataset is of relatively low quality, with a lot of spelling mistakes, misformed words and drug names and grammatically incorrect sentences. As an example, the following sentences were selected from the training set:

```
je prend minidrill et j'ai due oublier un comprimer et je ne l'ai pa
repris.  je voulais savoir si j'aret le mardi comme d'habitude ou si
je prend ts les comprimes ?


deroxat ??


Le L122 aide-t-il à bruler maigrir ?.
```

The first one contains some abbreviations such as `aret` for `arrête` or `ts` for `tous`. To deal with these types of mistakes, one will need to use a spell checker. These spelling errors might cause sparsity in the representation of sentences, with words such as `aret` that are only rarely present in the whole vocabulary although they has a strong meaning for the classification task at hand.

To perform spell checking on the dataset, I used the very fast spell checking algorithm SymSpell[1]. It uses as dictionary a word-frequency dictionary created from a dump of french Wikipedia from 2008. Since the total vocabulary of 3.2M words is too large for this task (most incorrectly spelled words are very close to a rare words, which adds bias to the correction), only the top 500000 words are used, augmented with a set of 2939 drug names scraped from Vidal's website[2]. The total number of words in the dataset went from 9542 words to 7973 with spell check. Additionally, some other pre-processing steps were applied to the text to eliminate some specific recurring patterns. These steps were

1. Strip all accents
2. Remove double dots patterns and replace with simple dots (`..` becomes `.`)
3. Split words of the form `xxx.xxxx` into `xxx . xxxx`
4. Split patterns of the form `xxxx/jour`
5. Tokenize sentences
6. Replace `l'xxx` by `xxx`

The purpose of this pre-processing is to make the dataset cleaner, with only real words in the vocabulary, without removing information from the sentences that could harm the performances of the algorithms down the line.

2 Model selection

I worked with several families of models that are able to capture different aspects of the data. Linear models are the most simple ones, while neural networks are more elaborate and have much more hyperparameters to tune. This makes it harder to find the best model and one often relies on best practises that might not be ideal for the task.

[1] `https://github.com/wolfgarbe/SymSpell`
[2] `https://www.vidal.fr/Sommaires/`

2.1 Linear models

The first family of models studied is linear models. To work with such models, sentences from the dataset need to be represented as vectors. The most obvious way to do so is to use bag-of-words (BOW) representations, where sentences are represented as histograms and the height of every bin corresponds to the number of occurrence of the corresponding word. This is illustrated on Figure 2

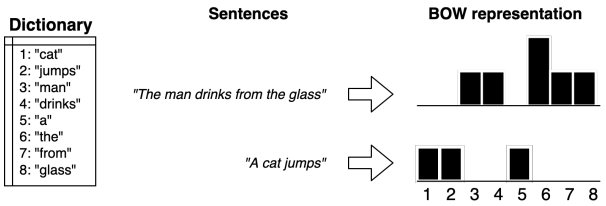


Figure 2: BOW representations of sentences

This representation doesn't capture any information about the sentence apart from the presence of a word and its frequency. With a large vocabulary, the BOW representation of sentences is usually very sparse and high dimensional, making it difficult to train linear models directly on these vectors for large corpora. For the dataset studied here, the total vocabulary is relatively small, with approximately tokens. A logistic regression can be fitted on this particular representation, yielding already an overall accuracy score of 60% on the subset of the 5000 most frequent words of the vocabulary without any hyperparameter selection.

The overall accuracy can be greatly improved by selecting the L_2 regularization parameter C of the regression by cross-validation and using tf-idf normalization the features. Tf-idf stands for term-frequency/inverse document frequency. This method consists in replacing the raw count (or frequency) of a word in BOW representation by a weight computed as follows

$$\text{tf-idf}(t, d) = f_{t,d} \cdot \log\left(\frac{N}{n_t}\right)$$

where $f_{t,d}$ it the term frequency in the document d (a sentence here) and n_t is the frequency of this word in the whole corpus. This representation adds extra weight to words that are frequent in a document but rare in the corpus and vice versa. Very frequent french words such as `le`, `et` have very low weights which corresponds to the intuition that they won't be very discriminative for classification. In practise, this representation has a positive effect on performance as Figure 3 shows.

This tf-idf representation used with a Logistic classifier gave as best performance an overall accuracy score of 68% on an unseen validation set. Since this comparative study was performed with a single train-validation set pair, there is a good chance that the selected model has a lower performance on the test set.

Although linear models might be very good at making complex prediction based on the presence/absence of words, they cannot handle order in the words (except when using it explicitly as a feature such as when adding n-grams to the vocabulary). Taking into accounts n-grams in the vocabulary has the drawback of making it very large even though there wouldn't be a need for two separate features to encode `my A` and `my B` if `A` and `B` have similar meaning. This is the kind of information word embeddings are capable to capture as explained in the corresponding section.

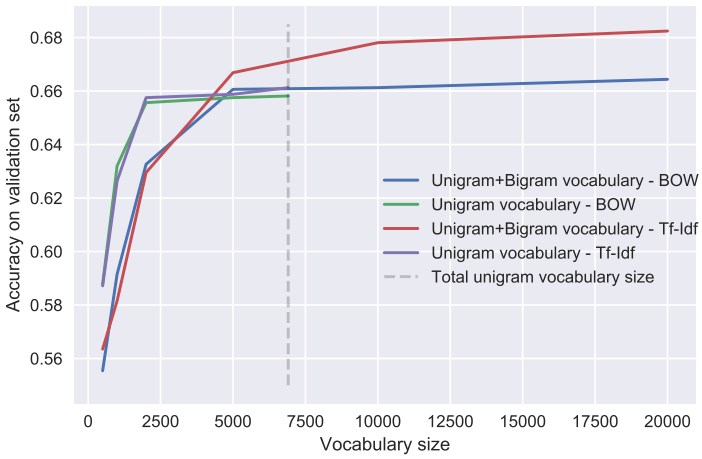


Figure 3: Performance scores the same algorithm with several sentence representations

2.2 Decision trees

Decision trees are able to cope with some of the problems mentioned above. The structure of these algorithms enables creating complex decision functions with features that seem very well adapted to text classification: decision trees can learn to make decisions based on the presence/absence of several words. I tried to train a decision tree based classifier and a random forest classifier that fits a large number of estimators on random subsets of the features and uses averaging to predict the final output. Random forest have good generalization properties and can deal with class imbalance relatively well.

One step of randomization further is the extra-randomized trees, that adds also a random component at the threshold selection level for splits in the estimator decision trees. Results of decision tree based algorithms on the dataset are one Figure 4.

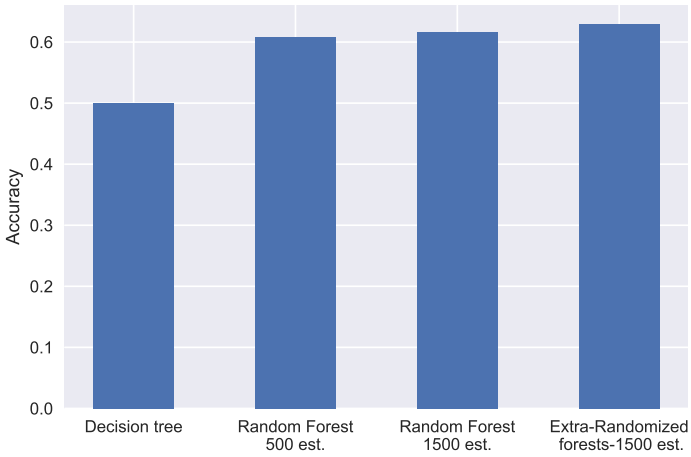


Figure 4: Performance of decision trees, random forests and extra randomized trees

2.3 Neural networks

The second class of algorithms I worked on are neural networks. These models usually deal better with complex non linear dependencies between input and output but are also harder to train and need a well specified architecture in order to effectively learn the structure of the data.

For example, it is not obvious that a multi-layer perceptron would achieve better results than a logistic regression with the previous data representation method. It appears in practise that it can approach it but I did not manage to make a MLP match the performance of the logistic regression.

Taking inspiration from the challenge organiser's benchmark description and Zhang and Wallace's paper [ZW15], I implemented a convolutional neural networks for classification. Both papers describe an architecture for text classification that consists in applying a small number of differently sized convolutional filters on a word embeddings representation of all the words in the sentences. A max-pooling operation is then applied on the whole length of the filtered sentence and fed into a fully-connected layer with softmax activation to predict the output probability.

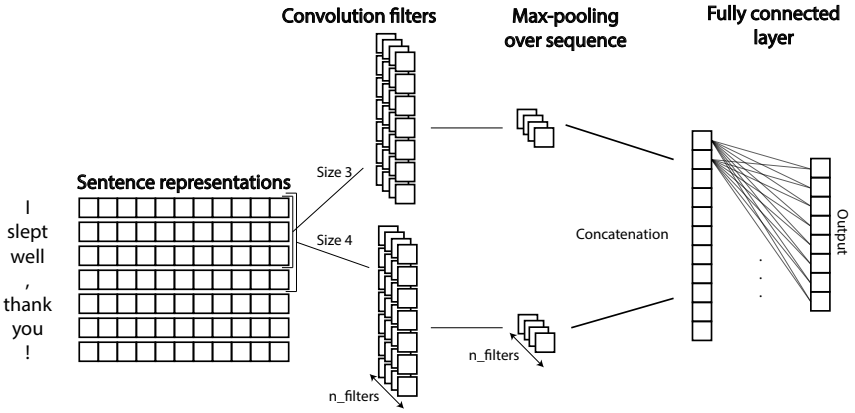


Figure 5: Convolutional network architecture for sentence classification

The intuition behind this architecture is to have "indicator" filters that are receptive to some particular sequence of words, which is going to be relayed by the max-pooling operation over the sentence. The last fully connected layer with softmax activation is essentially a linear classifier that is responsible for assigning the right class from the set of pattern sensitive layers.

There are two options for dealing with sentence representations,

- Work with pre-trained embeddings
- Use trainable embeddings and learn them during the learning process.

I tried both options, using the FastText pre-trained word embeddings for french from Grave et al. [Gra+18], and training some embeddings on the fly. Grave et al.'s embeddings are trained on Wikipedia dumps from 2017 with the CBOW model. Both gave similar results with my architecture as can be seen on Figure 6, with pre trained embeddings yielding a slight improvement over the trainable embeddings model.

However, although I tried to exactly match the specification from the challenge organiser's baseline model, the neural networks were nowhere close from achieving the advertised accuracy score of 82% accuracy on the test set. This might be caused by different pre-processing steps before running the algorithm.

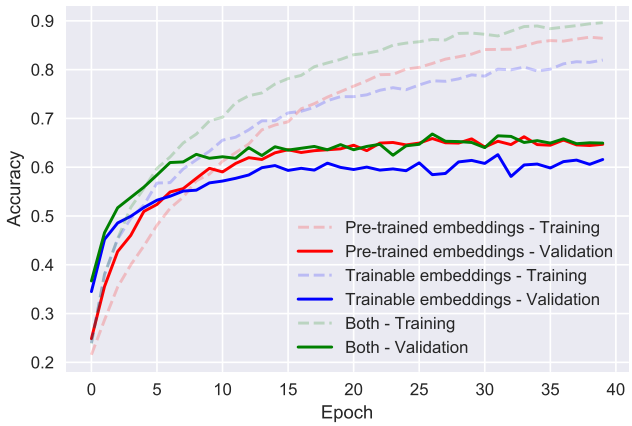


Figure 6: Comparison of different CNN based models with embeddings

3 Results and Error analysis

Results were overall not very satisfying, as it was very difficult to improve on the simple Logistic regression model and results with other models were disappointing. Some classes of the dataset are being confused with one another a lot by the classification algorithms, indicating that either the models weren't sufficiently well designed for capturing the discriminant information contained in the training sentences, or simply that the user created content of this dataset is sufficiently noisy to allow for some overlap between some of the 51 classes.

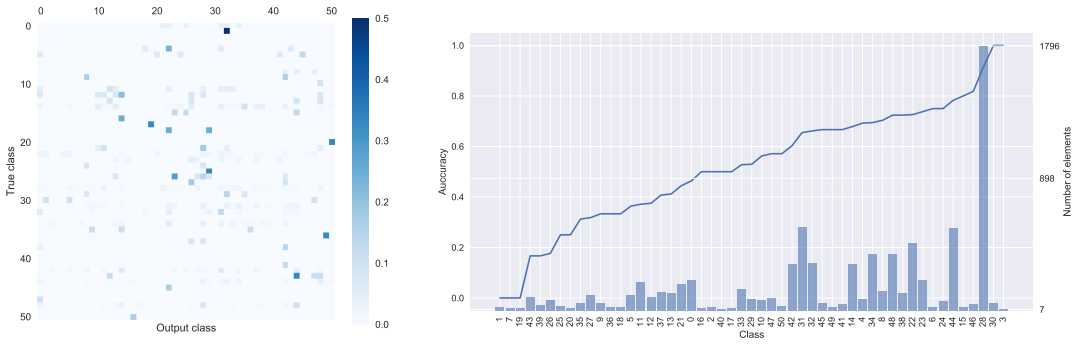
Errors can be inspected by looking at the confusion matrix of a classification algorithm. The confusion matrix is a matrix of size N-class$\times N$-class that contains for each row a count of the elements of the corresponding class that were classified as belonging to every other class. The confusion matrix of a perfect classification algorithm would contain elements on the diagonal only.

Figure 7a shows the confusion matrix of the best classifier I got. Many of the elements are just small confusions corresponding to mistakes made because of the noise in the inputs. However, it also appears that some classes are systematically mistaken for others, indicating high overlap between some of the classes of the dataset. This might either be due to real overlap (some classes have very similar definitions and some sentences can belong to several of them) between the classes or simply because the algorithms were optimized for the accuracy score and therefore focused on predicting the classes with most individuals.

Figure 1 shows the sorted class accuracies for the same model, along with the number of elements in the class. It appears clearly that the largest classes have highest accuracies while smaller classes aren't so well classified.

4 Possible improvements

- **Better embeddings**: use embeddings trained on medical NLP datasets (I couldn't find one in french). Better quality or domain specific embeddings might contain much more useful information for the classifier. For example, the word `pain` has very different meanings in a medical or general setting.
- Use **ensemble methods** with one ore more of the models presented above (they might have complementary strength). Random forest or extra randomized trees already are ensemble methods using decision trees as base methods. Other algorithms exist, such



(a) Confusion matrix for a logistic regression model

(b) Sorted accuracies per class

Figure 7: Error analysis for the linear model

as bagging and gradient boosting that make use of several small estimators to create a powerful ensemble estimator.

- With **class information** (either explicit class names of simple tf-idf on classes) one could get "easy wins" by designing specific rules when possible. In a practical setting such as the one the company POSOS is dealing with, class information is usually available because classes were designed by humans to serve a certain purpose (interaction between drugs, availability of a certain drug, etc.). Although manual design of rules isn't scalable at all, it might be the right tool for detecting very specific classes with well defined characteristics.

References

[ZW15] Ye Zhang and Byron Wallace. "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification". In: *arXiv:1510.03820 [cs]* (Oct. 13, 2015). arXiv: 1510.03820. URL: http://arxiv.org/abs/1510.03820 (visited on 03/22/2019).

[Gra+18] Edouard Grave et al. "Learning Word Vectors for 157 Languages". In: *arXiv:1802.06893 [cs]* (Feb. 19, 2018). arXiv: 1802.06893. URL: http://arxiv.org/abs/1802.06893 (visited on 01/17/2019).