Allard Dorian
Laeser Dorian
Leforestier Clément

Final Report NLP Project

Introduction

This project deals with toxic comments that can be found on the internet, especially on the website Wikipedia and how to classify them in different categories. Many classification problems only tackle binary labels such as positive/negative whereas for this project, the comments can be classified according to five labels: toxic, severe_toxic, obscene, threat, insult and identity_hate.

The comments are separated into a training set with 159 571 comments and a test set of nearly the same size with 153 164 comments, but not all of them were used in order to score the model.

As for the results, all the methods and observations are registered in a notebook that will be provided with this report. Some visualization graphs are plotted in the notebook. From the correlation matrix and from the individual histograms corresponding to each label, it can be observed that each comment can have different labels at the same time and that some labels are significantly correlated between each other.

Keras is the Deep learning library we used through this project for the models. We chose this one for its simplicity of use and efficiency. Also, a quick summary with the number of parameters trained and training time is automatically given at the end of the code thanks to this library.

Preprocessing

Before using the different classification models, some data cleansing was made on the dataset. For instance, we removed a list of stopwords (found online) from the comments, which are words that do not contribute to the global comprehension, they are mainly connexion words and this step helps to decrease the training time.

In addition, abbreviations were removed to facilitate the analysis, characters were lowered and also all other non ASCII characters were removed.

Then we tockenized each comment, meaning that we separated all the words present in the comments and put them in a nice list that will be used for training.

Each sentence was cut to length 150, and we considered the 100000 more frequent words in our train set to create our vocabulary. (The only exception is for CBOW, we used the 20000 more frequent words in the train set). We tried different values for those parameters and this combinaison gave us the best result.

Embedding

Finally we converted words into vectors, we trained each model twice (except CBOW), once with an embedding made from ourselves of size 300 whereas in the second time we used a collection of pre-trained vectors of size 300 coming from GloVe (840B tokens, 2.2M tokens) .

Models and results

Due to computation time and because it didn't improve a lot the result, we generally only used one epoch for training.

Results will be given firstly with our proper embedding and secondly with the one from GloVE.

All the models use the Adam optimizer coupled to the binary cross entropy loss and evaluate the performance with the accuracy metric. They also all have a dropout layer which is a simple trick to tackle overfitting and we set a 20% dropout.

We evaluated the model on the test set by submitting our prediction on the set to the kaggle competition, which use the mean column-wise ROC AUC as the metric

- The first model used to classify the comments is a Continuous Bag of Words (CBOW).
  - 45% accuracy was achieved and a loss equal to 0.1156 on the train set.
  - On 90% of the test set on kaggle, we obtained a score of 0.93616.

- The second model is a Convolutional Neural Network (CNN) with one convolutional layer. We tried several values for the number of filters (64, 128, 256) and the kernel size(3, 5, 7, 9), and we obtained our result by using 128 filters and a kernel size of 5.
  - 96,4% accuracy was achieved and a 0.0779 loss on the train set. We observed a huge improvement in accuracy. On 90% of the test set on kaggle, we obtained a score of 0.97308.
  - The second version slightly under performed on the train set but still gave a satisfying accuracy of 90%. With 90% of the test set on kaggle, we obtained a better performance with a score of 0.98083.

- The last model tried is the Long Short-term Memory mode (LSTM)l, which is a recurrent neural network, built with 2 LSTM layers.
  - 98.7% accuracy was achieved and a loss equal to 0.0889 on the train set. We observed a little improvement on the accuracy compared to the CNN model. On 90% of the test set on kaggle, we obtained a score of 0.96900, which is slightly lower in comparison with the score obtained with the CNN model.
  - With the pre-trained vectors, we obtained an accuracy on the train set of 96.6% and on the test set a score of 0.97627.

- Bonus : CNN + LSTM mixed, performed solely with tre pre-trained embedding. We also used a bidirectional LSTM layer, which takes the input sequence and the reverses order of the input sequence

    - We obtained an accuracy of 90% on the train set and a score of 98.2 on the test set.

Conclusion

CBOW is the simplest model and fastest model we implemented but it clearly under performs compared to the other two which gave more than 95% accuracy without pre-trained embedding and also with pre-trained embedding in the case of LSTM. If we consider the metric used during the kaggle challenge, the CNN model performs better than the LSTM model, and if we consider the accuracy on the train set the LSTM model performs better than the CNN model. Finally, the best performance on the test set was achieved with combinaison of LSTM and CNN layers. In order to improve the model and achieve better results, more time would have allowed us to try different architectures and values of hyperparameters (dropout values, token preprocessing).