Toxic Comment Classification
Machine Learning for Natural Language Processing 2020

Eléonore Blanchard
ENSAE Paris
eleonoreblanchard@gmail.com

Yannick Ly
ENSAE Paris
yannick.ly@ensae.fr

Abstract

Our work[1] aims to classify toxic comments. We have processed the texts of the comments using three different embedding vectorizers for sentences embedding (TF-IDF, Doc2Vec) and words embedding(Word2Vec). In order to improve our embedding we have used transfer learning to pre-trained our Word2Vec model. For the classification of the toxicity of the comments, we have used two classic model, logistic regression and light gradient boosting, which allow to make independent classification for each of the class of toxicity. Then, we have used an bidirectional LSTM model which is an artificial recurrent neural network which can process multi-class classification. Our model is tested on 63978 comments and we report a loss of .

1 Problem Framing

Due to the increase in the use of social media and the anonymity of internet, we observe a surge in the toxicity of comments and cyber-bullying. Some of the comments are event illegal because they are racists, homophobic or pedophile. This is why it is important that online platforms moderate this kind of comments and apply sanctions such as bans or the transmission of the personal data to authorities.
However, those platforms still have issues to correctly identify those comments, and more specifically their types of toxicity (toxic, severe toxic, obscene, threat, insult, identity hate).
Being able to identify the type of toxicity would make it easier to apply appropriate measures.
This subject appeal to us because we face this toxic behaviour on social media on a daily basis, and we are aware of the damage they can cause. Therefore we think it is important to find a solution in order to get rid of this toxicity on internet, in order to improve most users' experience.

[1] https://github.com/YannickLy/NLP-Project-ENSAE-2020

2 Experiments Protocol

2.1 Data

The data set that we are going to use come from Wikipedia comments. It is made up of 159 171 comments with 6 different labels (toxic, severe toxic, obscene, threat, insult, identity hate). They have been labeled by humans, therefore labelisation mistakes are unavoidable. Some of the comments have multiple labels.
The data available to us were only the comments and their labels, but no other features. This is why we have decided to build some features on the raw comments first in order to analyse them in respect of their length, aggressiveness and wealth of language.
Once these indirect features created, we have embedded the texts of our comments in order to use them in our further models. We have decided to use three different embedding vectorizers in order to compare them, and because models need different kind of embedding (words vs sentences embedding). Our three different embedders are :

- **TF-IDF** (Term Frequency-Inverse Document Frequency) : which evaluate the importance of a word in the document.
- **Word2Vec's Skip-Gram** (pre-trained on Google News' model) : which is built on words and try to find similarities.
- **Doc2Vec's PV-DBOW** : which is built on documents and their tags.

2.2 Models

The problem that we are facing in this project is a multi-labelled classification. Therefore, we are not only trying to predict one class for each comment, but rather to determine if a comment belongs to one or more of the six categories of toxicity (for instance a comment can be an insult and identity hate, but not a threat).
First of all we have used a logistic regression because we think it is appropriate to predict a probability for the comment to be in each category. It is a very straight forward approach that we wanted to use as our baseline.
We also have tried another approach for our baseline which consisted in training a light gradient boosting model, in order to make a classification.
Both of those baselines were implemented on each class of toxicity independently, which is not the most efficient approach, as shown by our metrics. Indeed, even if the loss are not very high, the confusion matrix can show that no matter which couple of model-embedding method we used, the true-positive rate were very low (which does not impact the loss a lot due to high unbalance in the classes).
This is why we have decided to use our final model: a Bidirectional LSTM (Long Short-Term Memory), which process all of our tags at the same time. The idea is to use a recurrent neural network with a soft-max activation function in order to make multi-class classification.
In order to implement this model, we need word embedding and this is where Word2Vec came particularly useful because it does not create an embedding vector for the whole comment but for the most important words in the comment. We then have implemented the model using Keras.

3 Results

The results (Figure 1) are pretty straight-forward. We can easily see that the bidirectional LSTM model works the best. The reason is threefold. First, it is suitable for our multi-class classification problem because it processes all the tags and takes into account the possible dependence in the toxicity classes. Second, a LSTM model holds in memory the past information and therefore can take into account the context of the comments. Third, the bidirectional LSTM is made of two neural networks (a backward one and a forward one) which allow it to hold in memory not only past, but also future information and therefore can predict the context of a word of both sides. This is why it is such a powerful model compared to the two classic baselines that we have implemented.

Modele	Sans embedding	TFIDF	Word2Vec	Doc2Vec
Regression logistique	0.133352	0.107077	0.093664	0.0792287
LightGBM	0.13035	0.113314	0.113422	0.104847
Bidirectional LSTM	NaN	NaN	0.075314	NaN

Figure 1: Comparison of our models.

4 Discussion/Conclusion

To improve our model we could have used a BERT model which is a framework created by Google which give general good results and could be a good benchmark to compare the bidirectional LSTM. Moreover, we could also improve the bidirectional LSTM by fine tuning its parameters. When it comes to the embedding models we could try to improve our Doc2Vec model by using a PV-DM model and compare it with the one that we used (PV-DBOW) to see which one works best. We also wanted to improve it by using a pretrained model but we did not find any satisfactory data.