NLP project: Toxic Comment Classification

Marzook Cyril & Reberga Louis

Friday 15 April 2022

Dauphine UNIVERSITÉ PARIS | PSL

Contents

1 Introduction **3**

2 Prepocessing **3**

3 Embeddings: GloVe **4**

4 Models **4**

4.1 CNN 4

4.2 LSTM bidirectional LSTM 4

4.3 GRU bidirectional GRU 5

5 Results comparison **5**

6 Conclusion **6**

7 Bibliography **7**

1 Introduction

This project is based on a Kaggle competition proposed by Jigsaw, a Google's subsidaries belonging to Alphabet. In this competition, the challenge was to build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's models in 2018. Perspective is an API developed by the Conversation AI team, a research initiative founded by Jigsaw, using machine learning to reduce toxicity online.
For us, the challenge is to use and implement the different classification methods we saw during the NLP courses. We decided to use the pretrained embedding **GloVe 6B** with 4 different vector dimmensions: 50, 100, 200 and 300. We will try all of them and compare the results. For the model, we decided to try 5 different architectures: CNN, LSTM, bidirectional LSTM, GRU and bidirectional GRU to compare the results and find the best architecture for this classification task.

2 Prepocessing

Before using our models to learn and predict the toxicity of the comments we have to prepocess them. There are few steps to implement:

1 - **Comments cleaning**: There are three parts in the comments cleaning. First, we have to remove all the capital letters. Indeed, for a model *heLLo*, *Hello* and *HELLO* are not the same word however these words have the same meaning. So, this is very important to have only lower cases in our comments. Then, we have to remove all the special caracters. Indeed, these special caracters have no real meaning in our context and for our classification task. Finally, we have to remove all the stop words. The most common stop words are pronouns, articles, prepositions, and conjunctions. This includes words like a, an, the, and, it, for, or, but, in, my, your, our, and their. They have no meaning neither and are useless to understand the sense of the comments and to classify them.

2 - **Tokenization**: Now, we have cleaned comments. The tokenization split the comments into a list of words. Then, each word is indexed and the result of the tokenization is a vector of integers.

Ex: "I love this video" → ["I", "love", "this", "video"] → [1, 254, 488, 96]

3 - **Padding**: The last part of the preprocessing is the padding. Models can only take as input same size vectors/matrices. So, we have to fill the tokenized vectors with 0 if the comment is to short. The size of the vector input is chosen as hyperparameter. We decided to use a post padding, the vector is filled at the end. We could also choose to pad comment before or in the both side.

Ex: If the max size chosen is 8: [1, 254, 488, 96] → [1, 254, 488, 96, 0, 0, 0, 0]

3 Embeddings: GloVe

GloVe, coined from Global Vectors for Word Representation, is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words. GloVe is an alternate method to create word embeddings. It is based on matrix factorization techniques on the word-context matrix. A large matrix of co-occurrence information is constructed and you count each word as the rows, and how frequently we see this word in some context corresponding to the columns in a large corpus. Usually, we scan our corpus in the following manner: for each term, we look for context terms within some area defined by a window-size before the term and a window-size after the term. Also, we give less weight for more distant words. The number of contexts is, of course, large, since it is essentially combinatorial in size. So then we factorize this matrix to yield a lower-dimensional matrix, where each row now yields a vector representation for each word. In general, this is done by minimizing a "reconstruction loss". This loss tries to find the lower-dimensional representations which can explain most of the variance in the high-dimensional data.

4 Models

For the model, we decided to try and compare multiple architectures. The architectures we tried are CNN, LSTM, bidirectional LSTM, GRU and bidirectional GRU.

4.1 CNN

A convolutional neural network (CNN) includes successively an input layer, multiple hidden layers, and an output layer, the input layer will be dissimilar according to various applications. The hidden layers, which are the core block of a CNN architecture, consist of a series of convolutional layers, pooling layers, and finally export the output through the fully-connected layer. The convolutional layer is the core building block of a CNN. In short, the input with a specific shape will be abstracted to a feature map after passing the convolutional layer. a set of learnable filters (or kernels) plays an important role throughout this process. The following Figure 5.2 provides a more intuitive explanation of the convolutional layer. We have also improved convolutional layer which was proposed by Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oor,d Alex Grave and Koray Kavukcuoglu and this kind of convolution is named Dilated convolution, in order to solve the problem that the pooling operation in the pooling layer will lose a lot of information. The critical contribution of this convolution is that the receptive field of the network will not be reduced by removing the pooling operation. We used the relu and sigmoid activation functions.

4.2 LSTM bidirectional LSTM

Long Short Term Memory Network (LSTM) is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. In Figure 1, you can see a schema explaining the architecture of the LSTM module. In bidirectional, our input flows in two directions, making a bi-LSTM different from the regular LSTM. With the regular LSTM, we can make input flow in one direction, either backwards or forward. However, in bidirectional, we can make the input flow in both directions to preserve the future and the past information. For our classification task, the bidirectional LSTM architecture gets better scores than classical LSTM.

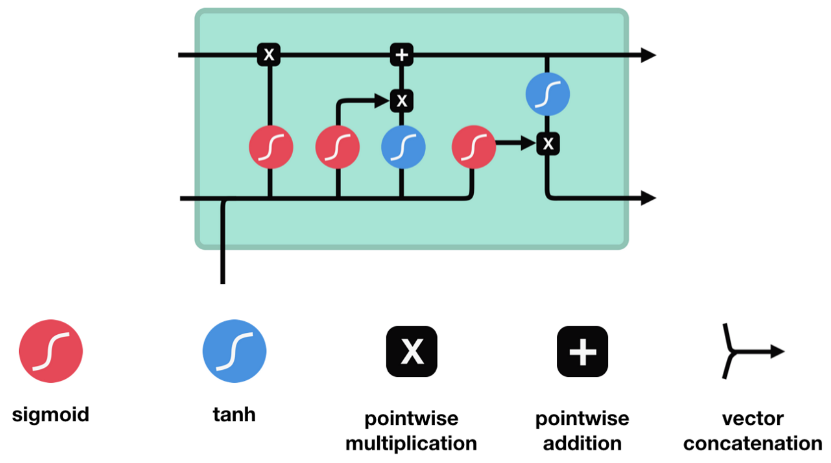

Figure 1: Long Short Term Memory (LSTM)

4.3 GRU bidirectional GRU

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM. GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets. That is why we have choosen this model for this project. Bidirectional GRU's are a type of bidirectional recurrent neural networks with only the input and forget gates. It allows for the use of information from both previous time steps and later time steps to make predictions about the current state. So a Bidirectional GRU, is a sequence processing model that consists of two GRUs. one taking the input in a forward direction, and the other in a backwards direction. It is a bidirectional recurrent neural network with only the input and forget gates.

5 Results comparison

We are now comparing all the methods listed above. Let's begin with GloVe 50D. One can observe that the more efficient method is the bidirectional LSTM followed by the bidirectional GRU with respectively 0.97389 and 0.97373 in accuracy. Then for GloVe 100D, it is pretty much the same situation where bidirectional LSTM and bidirectional GRU perform better. Moreover, for GloVe 200D we can see that bidirectional GRU performs even better than bidirectional LSTM with an

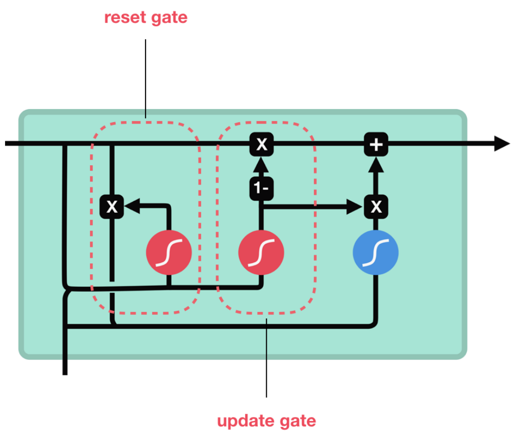

Figure 2: Gated Recurrent Unit (GRU)

accuracy of 0.97828. The higher accuracy is achieved with GloVe 300D. The method used to get this accuracy is again bidirectional LSTM. Overall all differents types of method and embeddings, the best combinaison is a GloVe 300D with a bidirectional LSTM. With this settings we achieved a 0.97898 accuracy which is really impressive and it outperformed our thoughts. We can also observe that for all GloVe types of embeddings, classic convolutional neural network is the least effective method with the lower accuracy.

	CNN	LSTM	Bidirectional LSTM	GRU	Bidirectional GRU
GloVe 50D	0.96677	0.97146	0.97389	0.9712	0.97373
GloVe 100D	0.97205	0.97415	0.9778	0.97322	0.97742
GloVe 200D	0.97147	0.97599	0.97805	0.97702	0.97828
GloVe 300D	0.97249	0.97612	0.97898	0.97669	0.97855

6 Conclusion

During this project we have tested and implemented five differents methods, convolutional neural network, Long Short Term Memory Network (LSTM) and bidirectional LSTM and finally Gated Recurrent Units (GRU) and bidirectional GRU. Beyond all those methods, we used different types of embedding with GloVe (50D to 300D). After a carefull analyze of our results we can conclude that the best model for our problem which is detecting different types of toxicity like threats, obscenity, insults, and identity-based hate, is a bidirectional LSTM.

7 Bibliography

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oor,d Alex Grave ans Koray Kavukcuoglu, (2016) *Neural Machine Translation in Linear Time.*
`https://arxiv.org/pdf/1610.10099.pdf`

Cho, Kyunghyun, van Merrienboer, Bart, Bahdanau, DZmitry, Bengio, Yoshua (2014) *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.*
`https://arxiv.org/abs/1409.1259`