Recurrent neural networks for newsgroup documents classification

Asier Barrasa and Julen Azpiroz

Abstract

Recurrent neural networks have been one of the most extensively applied methods to text processing. On this project, we have implemented a LSTM, RNN-based neural network for classifying text across 20 different classes with Keras. In order to get better results, we have used Glove[1] and Fasttext[2], an unsupervised learning algorithms that obtains vector representations for words, in stead of using TFIDF or count vectorized. We have used, *keras* for making the network, *numpy* for some maths and arrays, *os* and *sys*, for reading directories, *contractions* for expand contractions and *sklearn* for split the data on test and training and making the confusion matrix.

1 Description of the problem

The goal of this project is to classify news across 20 different classes. The database used in this project is 20 Newsgroup [3], the database have nearly 20.000 news from 20 different topics.

The database has the following characteristics:

- 20 classes, almost balanced.
- 18, 828 instances.
- An average of 218 words per text.

2 Description of our approach

We organized the implementation of the project according to the tasks:

1. Preprocessing of the dataset.
   (a) Process the directories.
   (b) Clean the data
2. Process data in order to get embedding matrix.
   (a) GloVe [1]
   (b) Fasttext [2]
3. Design and run the NN.

2.1 Preprocessing

We cleaned each document, removing the header of each text because it does not get any information,and expanding all texts contraction with a library called contractions[4].

[1] https://nlp.stanford.edu/projects/glove/
[2] https://fasttext.cc/
[3] http://qwone.com/ jason/20Newsgroups/
[4] https://github.com/kootenpv/contractions

2.1.1 Directories and classes

This was probably one of the largest part of our project. Because of the amount of archives. Firstly be had to assigns to each class a number between 0 and 19, and to the document on each directory the corresponding number.

After carrying out this process, we realized that it was a balanced problem, there were almost the same number of data of each class.

$0 : 799, 1 : 973, 2 : 985, 3 : 982, 4 : 961, 5 : 980, 6 : 972, 7 : 990, 8 : 994, 9 : 994, 10 : 999, 11 : 991, 12 : 981, 13 : 990, 14 : 987, 15 : 997, 16 : 910, 17 : 940, 18 : 775, 19 : 628$

2.2 GloVe

To get the embedding matrix, we tokenized all the documents, in order to get a dictionary with all the different words and a number based on the number of appearances. After this, we fit all the texts in order to replace each word with its corresponding number and fit the length of each text.

Once we have the most common 20,000 words, we made the embedding matrix,

We look for the min of 20.000 and the total different number of words, in case there were less than 20,000. As we have $153,989$ different words, so we have only chosen the $20,000$ most frequent words for the problem.

The embedding matrix shape is $(20,000 \times 100)$.

100 is the length of the GloVe vector.

2.3 Fasttext

Fasttext (which is essentially an extension of word2vec model), treats each word as composed of character ngrams. So the vector for a word is made of the sum of this character n grams.

This makes Fasttext better with unknown and rare words and improves the performance.

The embedding matrix shape is $(20,000 \times 300)$.

300 is the length of the Fasttext vector.

3 LSTM and CuDNNLSTM

LSTM(Long short term memory) networks, are a special kind of RNN, capable of learning long-term dependencies. This kind of networks, performs very well with text.

For this proyect we have used, the NVIDIA CUDA Deep Neural Network library (cuDNN[5]), witch is a GPU-accelerated library of primitives for deep neural networks.

Using the CuDNN and a NVDIA GeForce GTX 960 the perform of the network improves significantly.

[5] https://developer.nvidia.com/cudnn

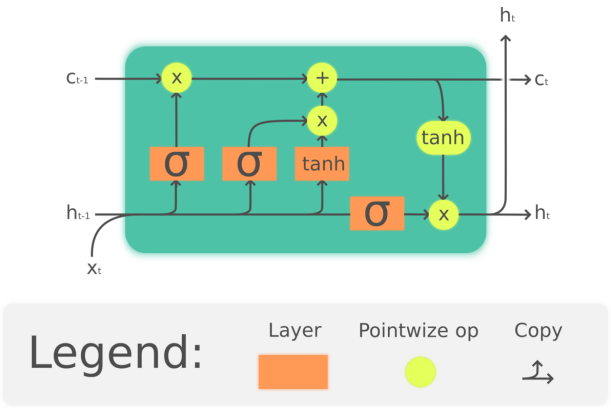

Figure 1: LSTM unit

4 Classification

For the classification of the news, as we say, we implemented a recurrent neural network with LSTM neurons on the second layer. We have tried with 128 and 256 neurons on this layer.

The first layer is a layer with the confusion matrix weights, with an input of 1000.

The third layer has a relu activation, we have tried with 128 and 256 neurons, the best results have been obtained with 128 neurons.

For the last layer, we have used a softmax activation.

The optimization function, we have tried with both adam and rmsprop.

The best results were obtained with 128 LSTM neurons, 128 neurons in the third layer and activation type relu, and adam optimization function.

5 Results

We have made several tests, and we have been testing with different amounts of epochs, from 15 to 30. The model converges in the epoch 30. The highest accuracy obtained was 67,85%

Although Fasttext in general has a better performance, the best results were obtained with GloVe, and the optimization function adam with 128 neurons in each layer.

Embedding Vectors	LSTM	Dense	Optimization Function	Epoch Number	Accuracy
Fasttext	128	256	rmsprop	30	63.02%
Fasttext	128	128	rmsprop	30	64.85%
Fasttext	128	256	adam	30	63.98%
Fasttext	128	128	adam	30	64.28%
GloVe	128	256	rmsprop	30	62.28%
GloVe	128	128	rmsprop	30	62.38%
GloVe	128	256	adam	30	65.30%
GloVe	128	128	adam	30	**67.85%**

Table 1: Results table

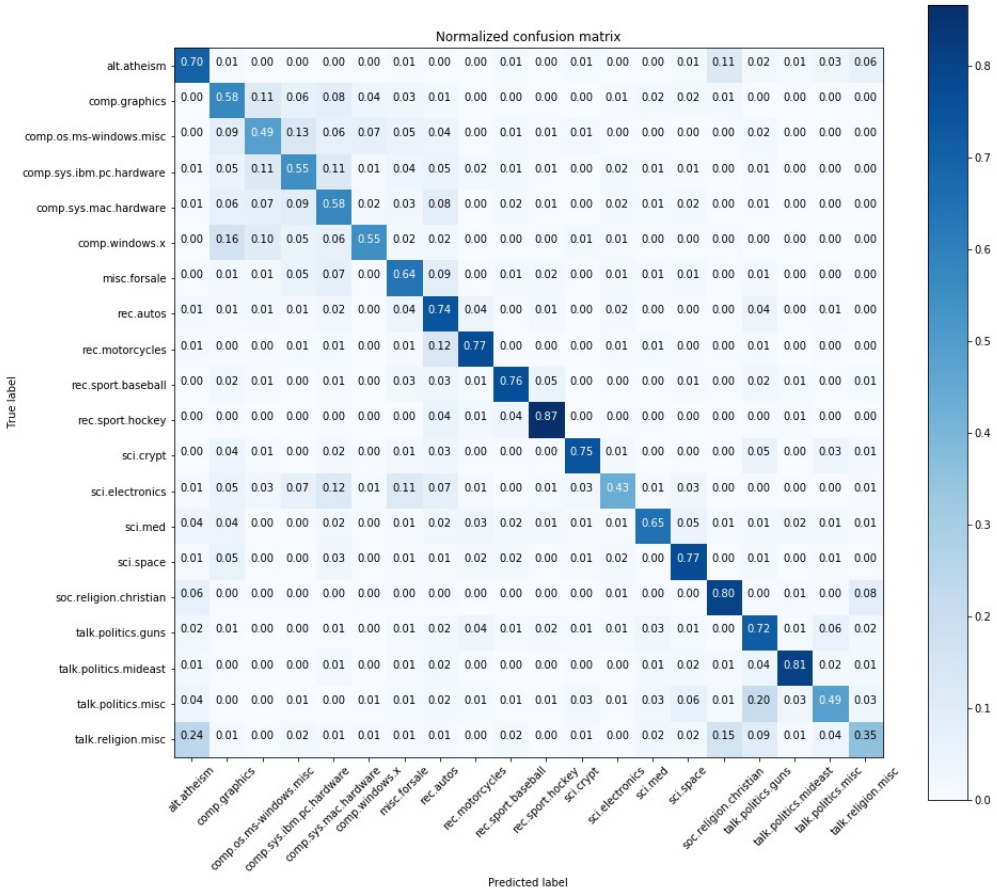

Figure 2: Confusion Matrix

6 Conclusions

Even the results are not bad, we think that results could be better if we have used other technique like TF-IDF or Count Vectorized.

References

[1] JeffreyPennington, RichardSocher, ChristopherD.Manning [*GloVe: GlobalVectorsforWordRepresentation*]. https://nlp.stanford.edu/pubs/glove.pdf

[2] PiotrBojanowski and EdouardGrave and ArmandJoulin and TomasMikolov [*Enriching Word Vectors with Subword Information*]. https://arxiv.org/pdf/1607.04606