Sentiment Analysis using BiLSTM and CNN

Aristeidis Noulis
University of Trento
aristeidis.noulis@studenti.unitn.it

December 12, 2019

1 Introduction and Objectives

Sentiment analysis or opinion mining is the computational study of people's opinions, sentiments, emotions, appraisals, and attitudes towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes. The objective of this project was to implement sentiment analysis on human written movie reviews, using deep learning models. To determine whether the given movie review has a positive or negative sentiment, two different models were developed, one BiLSTM and one CNN. At the beginning, simple versions of them were established and used as baseline models. Subsequently, a number of modifications were tried and added on top of these baselines, in order to achieve better accuracy on the classification of every review as positive or negative. In this paper, the detailed procedure of development and experimental phases will be presented, followed by the final conclusions and results.

2 Models Overview

2.1 Implementation Environment

Both of the models of the project have been implemented in Python with the use of Pytorch, a machine learning library and Google Colab, a free Jupyter notebook environment.

2.2 BiLSTM

The first model that has been developed is a Bidirectional Long Short-Term Memory (BiLSTM) neural network, which combines two LSTMs, one forward and one backward. LSTMs overcome the problem of the standard RNNs, which suffer from the vanishing gradient problem, by having an extra recurrent state called cell c (which can be thought as the "memory" of the LSTM). Moreover it uses multiple gates which control the flow of information into and out of the memory.

At time-step t, the memory c_t and the hidden state h_t of the LSTM are updated with the following equations:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ g_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t] \tag{1}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2}$$

$$h_t = o_t \odot \tanh c_t \tag{3}$$

where x_t is the input at the current time-step, i, f and o are the input gate activation, forget gate activation and output gate activation respectively, g is the current cell state, σ denotes the logistic sigmoid function and $\odot$ denotes element-wise multiplication.

The LSTM can be expressed as a function of x_t, h_t and c_t according to the following formula:

$$(h_t, c_t) = \text{LSTM}(x_t, h_{t-1}, c_{t-1}) \tag{4}$$

The concept behind the implemented BiLSTM baseline model can be analyzed in the following steps:

- The network is being fed with the no pretrained word embeddings of the sentence.
- The forward LSTM processing the words in the sentence from the first to the last, and the backword LSTM processing the words from the last to the first.
- Then the concatenation is performed. The last hidden state from the forward LSTM (obtained from final word of the sentence), h_{f_T}, and the last hidden state from the backward LSTM (obtained from the first word of the sentence), h_{b_T}, are concated giving a prediction of this form:

$$\hat{y} = f(h_{f_T}, h_{b_T}). \tag{5}$$

- Finally the result is passing through the linear layer.

The figure 1 visualizes the operation of the BiLSTM baseline model, with the forward LSTM in orange, the backward LSTM in blue and the concatenation and linear layer in silver.

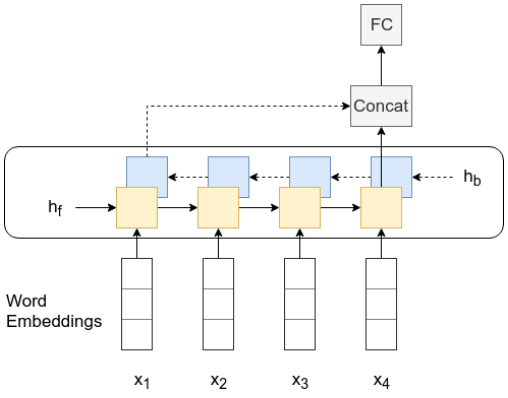

Figure 1: BiLSTM Baseline

The model has been initialized with 128 hidden states dimension, and the modifications on this field will be analyzed more on the experiments chapter.

2.3 CNN

Convolutional Neural Network (CNN) is a special type of feedforward neural network originally employed in the field of computer vision. However some modification can allow it to be used also for text classification. The one that has been implemented in this project is based on the paper of Yoon Kim, Convolutional Neural Networks for Sentence Classification [2].

At the beginning 3 convolutional layers of dimension 2 have been implemented. Normally, the input channels of convolutional layers are 3, as the images need one for each of the color channels (red, blue, green), whereas for text there is one single channel. Following the paper that it is mentioned before [2], three different filters of height 3,4,5 with 100 feature maps each have been used. For this reason the number of the output channels for each convolutional layer is 100. Similarly the kernel size number following the rule of $[h \times emb-dim]$ where h corresponds to each filter height and $emb-dim$ to the dimensionality of the used embedding vector.

The workflow in the CNN can be describes as follows:

- The word embeddings elements are passing to convolutional layers, and the RELU activation function is being used after that.
- In next step, the tensors pass to Max Pooling layers. The max pooling operation is applied with goal to take the maximum value of the feature map, which corresponds to the particular filter. The idea is to capture the most important feature (the one with the highest value) for each feature map. This pooling scheme naturally deals with variable sentence lengths.
- Then the concatenation is performed on the filters outputs.
- Finally the concatenated result is passing to the linear layer.

The figure 2 presents how the CNN baselines is working.

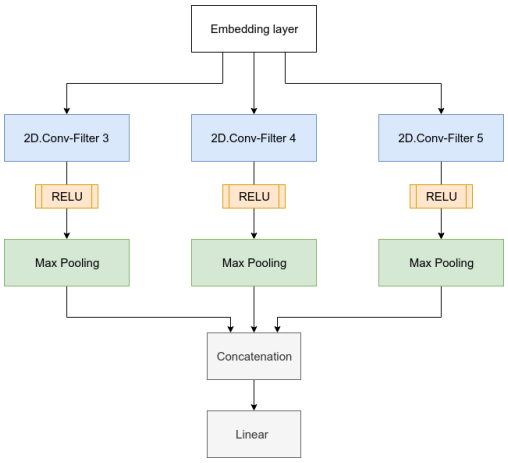

Figure 2: CNN Baseline

restricted to a number between 0 and 1. This can be done by the sigmoid function. After that, this bound scalar is being used to calculate the loss using binary cross entropy. The BCEWithLogitsLoss criterion carries out both the sigmoid and the binary cross entropy steps and for this reason has been selected for the loss function.

The accuracy of the models is calculated in a binary way. A function has been used to round any prediction value greater than 0.5 to 1 (a positive sentiment) and the rest to 0 (a negative sentiment). After that, the fuction calculate how many rounded predictions equal the actual labels and averages it across the batch.

3 Dataset and Training setup

3.1 Dataset

The Dataset that has been used, is from the Standord webpage and is allowed to be used for educational purposes [4],[3].

The Dataset contains 50.000 movie reviews, 25.000 of them for training and 25.000 for testing. Both of the subdatasets are equaled divided in positive and negative reviews.

The training subdataset, splitted in 17.500 reviews for the training part of the neutral networks and 7.500 reviews that have been used to evaluate the models.

The label for the positive reviews has been set up to 1 and for the negative ones to 0, for reasons of flexibility.

3.2 Training

For both of the models the Adaptive Moment Estimation (Adam) optimizer has been used. Adam adapts the learning rate for each parameter, giving parameters that are updated more frequently lower learning rates and parameters that are updated infrequently higher learning rates.

As the models output an unbound real number and the labels are either 0 or 1, the predictions should be

4 Experiments and Results

4.1 Hidden Dimensions

This experiment phase concerns only the BiLSTM model and modificates the number of its hidden states. The baseline model has been initialized with 128 hidden dinmensionality. In the experiments have tried also the dimension 256 and 512.

The number of layers and cells required in an LSTM might depend on several aspects of the problem:

- The complexity of the dataset. The amount of features, number of data points etc.
- The data generating process. Following example of how data generating process can play significant part.
- The accuracy required for the use case.

In this project tested the performance of the BiLSTM in every hidden dimension, respected to time and to accuracy and selected the best one.

According to the table 1, the best score is with hidden dimension 256, so this modification is added to the model and the next experiments will be tested on this improved version of the BiLSTM model.

Table 1: Hidden States Results

Hidden States List	
Modifications	BiLSTM Score
Hidden 128	82.56
Hidden 256	85.83
Hidden 512	84.84

Table 2: Glove Results

Glove Results List		
Modifications	BiLSTM Score	CNN Score
Glove 100	85.98	86.74
Glove 200	85.51	87.04
Glove 300	87.43	87.23

4.2 Glove

When applying one-hot encoding to words, the result is ending up with sparse (containing many zeros) vectors of high dimensionality. On large datasets, this could cause performance issues. Additionally, one-hot encoding does not take into account the semantics of the words. So words like "movie" and "film" are considered to be two different features. While they have a very similar meaning. Word embeddings address these two issues.

Word embeddings are dense vectors with much lower dimensionality. Secondly, the semantic relationships between words are reflected in the distance and direction of the vectors. The theory behind is that these pretrained vectors already have words with similar semantic meaning close together in vector space, as words like "terrible", "awful", "dreadful" are nearby.

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Glove vectors are count-base vectors.[1]
Using Glove vectors gives to the embedding layer a good initialization as it does not have to learn these relations from scratch. Sadly, words which are not present in the set of pretrained words are initialized randomly.

The Glove vectors that have been used are of 100, 200, 300 dimensions, which are publicly available and trained on 6 billion tokens.

The Glove vector with the best accuracy result selected for each model. As the table 2 shows, for both of the models this is the Glove of 300 dimensions.

4.3 English StopWords Removal

English StopWords are the English words which do not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For example, the words like the, he, have etc.

However, in the context of sentiment analysis, removing stop words can be problematic if the context is affected and its sentiment can be changed. One case like this is when the StopWords corpus includes the word 'not', which is a negation that can alter the valence of the passage.

The reason why the removal of the StopWords has been tried in this project was to lower the dimensional space and not let few stop words drive the analysis.

4.4 Regularization

The modifications that can be added to the models, will increase the additional parameters. In other words, the probability of an overfitted model will be higher. This mainly happens because memorizing the training data, may cause a low training error but validation/testing error is higher, (poor generalization to new, unseen examples in other words).

To combat this situation, the dropout method has been used. Dropout works by randomly dropping out (setting to 0) neurons in a layer during a forward pass. The probability that each neuron is dropped out is set by a hyperparameter p and each neuron with dropout applied is considered indepenently. In this experiment the $p = 0.5$.

4.5 Results Table

The table 3 shows the final experiments results. Every modification is added to the previous successful one, so an order from up to down is followed. In other

words, an improved is tested and if it achieves better accuracy then it is added to the model.

Table 3: Results

Results List		
Modifications	BiLSTM Score	CNN Score
Baseline	82.56	86.55
Hidden 256	85.83	
Glove 300	87.43	87.23
StopWords	87.93	87.71
Dropout	88.61	88.38

5 Conclusions

This project presents two different models for sentiment analysis, one is the BiLSTM and the second is a CNN. The experiments results of the Table 3 shows in a clear way that all the improvements which have been added, helped the models to achieve better accuracy scores.

The improvement of BiLSTM through modifications was bigger in comparison with CNN, but in both of the cases the final models achieved the higher results.

There is still a lot of space for improvements, as more experiments can be done on parametrizing the models.

However, quite interesting are the combinations that can arise by mixing the models in both directions (CNN-BiLSTM and BiLSTM-CNN). One example can be the integration of the BiLSTM with a 2 dimensional convolutional layer and a 2 dimensional max pooling layer, as this operation over the two dimensions may sample more meaningful features.

Furthermore, the use of a pretrained word embedding for movie reviews vocabulary may lead to higher results.

References

[1] Christopher D. Manning Jeffrey Pennington, Richard Socher. Glove: Global vectors for word representation. `https://nlp.stanford.edu/projects/glove/`, 2014.

[2] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[4] Stanford. Stanford large movie review dataset. `http://ai.stanford.edu/~amaas/data/sentiment/`, 2011.