DEEP LEARNING

MASTER IN ARTIFICIAL INTELLIGENCE

Universitat Politècnica de Catalunya

PRACTICAL WORK 1:

CONVOLUTIONAL NEURAL NETWORKS APPLIED TO POLARITY CLASSIFICATION IN IMDB MOVIE REVIEWS

Albert Espín

March 5th, 2019

Table Of Contents

1. Introduction

The Internet Movie Database (IMDB) movie reviews data set is a well-known collection of plain-text film reviews, labeled as positive or negative (polarity of the opinion). Convolutional Neural Networks (CNNs) are a type of neural networks where filters are applied to groups of adjacent data elements to detect local paterns, which can be useful in the field of text analysis to detect associations of words in sentences.

The aim of this work is to train a deep learning model with CNNs to classify the polarity of the reviews with high accuracy (as close as possible to the state of the art), analyzing and evaluating the architecture, trying to refine it with different experiments. The develop code is available at [1].

2. The data set

2.1. Data representation

The data set contains 50000 instances representing reviews, from which half (25000) are labeled as positive and the other half are negative ones: these are the only two classes of data. The reviews use a total of 9998 unique different words.

Whilst the original version of the IMDB movie reviews data set contains every review as text in natural language, this work uses the Keras version of the data set, in which every review is a list of the indices of the words that it contains. Each index is an integer number, unique for each word: a word has the same index in any review where it appears. The index also indicates the popularity of the word, i.e. words are sorted by their number of occurrences in the data. The most popular word has index 1, since 0 is reserved for unknown or unclassified terms.

A key factor of the data set is the length of the review texts. The mean length is 234.759 words, with a standard deviation of 172.913, which indicates that the variability among the length through different reviews is not very big: 90% of the texts have between 92 and 457 words. A small percentage of reviews is longer: 5% have more than 598 words, while only 0.01% contain more than 1034 terms; the longest review contains 2494 words, significantly longer than the shortest, which has only 7 terms. The boxplot represented in Figure 1 depicts the data distribution.

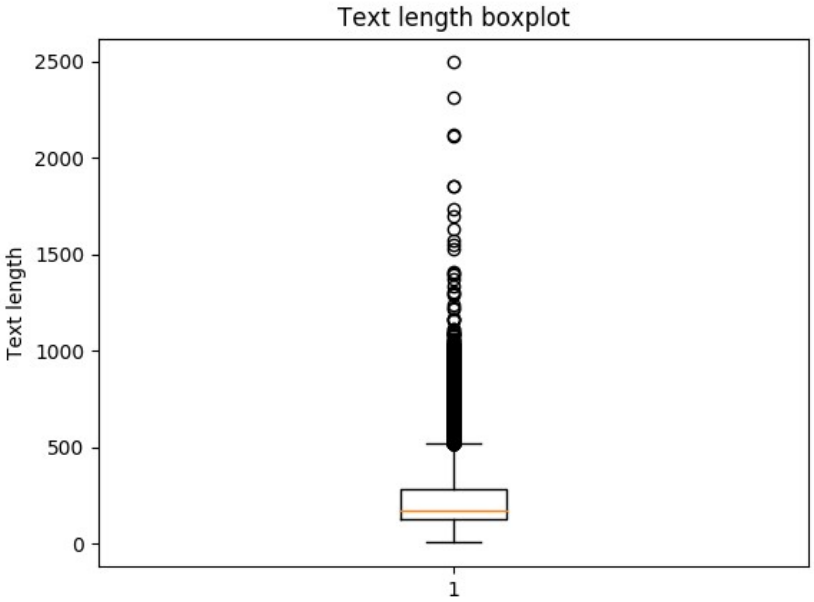

Figure 1: Boxplot showing the distribution of the length of the reviews.

2.2. Preprocessing

To be able to apply CNNs and filters to the indexed reviews, it is required that all of them have the same, fixed length. This is achieved with the padding technique: texts longer than the fixed length are truncated, while shorter ones get 0 values added at the end, until the desired length is achieved. Recall that in the data notation, 0 represents a missing or unknown value, so the model will distinguish it from the rest of words. This task requires to decide the fixed length of texts, that has been selected with the will to preserve as much information as significantly possible, while also saving memory space for the least relevant data. After analysing the distribution of the length of the texts, one can see that 99.5% words contain less than 1000, so this has been set as the fixed length, which only truncates a very reduced number of instances, that have up to 2494. Avoiding truncation would imply using more than twice memory space that would be used only be the 0.05% of the reviews.

The other important decision regarding preprocessing was whether some words should be ignored. One could think that removing the least popular words or the

most popular ones (commonly stop-words, i.e. words like "the" or "a", with little semantic information) may yield better results, but the tested experiments show the contrary, with accuracy decreasing roughly between 2% and 4% if the top 15 most popular words is removed, or if the model ignores the least 30 popular words. This can be explained by the textual analysis performed by the model, that emphasizes the local association of words with the usage of CNNs. If we eliminate words that by themselves may have little meaning contribution, we are also losing contextual information that the model may use to learn (e.g. it is not the same "the good film" and "a good film"). The same reasoning applies for uncommon, least popular words (e.g. if "unbearable" was uncommon, it is not the same "the movie" and "the unbearable movie").

2.3. Data splits

The data has been divided in three splits: training (to train the model), validation (to select the best model configuration) and test (to assess the accuracy of the final model). Before performing the split, the data has been shuffled, to avoid any potential bias due to the ordering of the instances (i.e. imagine that the instances were sorted by label, or by text length).

Since there are only two classes of data, it has been considered that 1000 reviews should be enough to capture a sufficient quantity of instances of each class (approximately 500 per class), of a variety of text lengths. For this reason, both the validation and test sets have been given 1000 reviews (2% of the data) each, while the remaining 48000 instances have been used to train the model (96%).

3. The model's architecture

3.1. Basic architecture

This section presents the layers that make up a reasonably simple neural network model used to solve the polarity classification problem. All the layers are stacked up sequentially, in the following order:

1. Input layer: contains the 1000 word indices of a review's text.
2. Embedding layer: projects the word indices into a vector space, obtaining dense vectors. Through the training process, the model learns to place closely in that space the dense vectors of words that are related in the polarity classification process. The dimension of the vector space has been set to 60,

since it improved the results over the default 32 option by approximately 1.5%. This may be explained by the fact that, with almost 1000 different words in the vocabulary, more dimensions may allow to represent the relationships of words in a more exhaustive, detailed way.

3. 1-D convolution layer: applies 300 filters to learn local association relations of word embeddings. The quantity of filters may seem large considering that there are at most 1000 words per text, but smaller quantities, e.g. 100, decreased the accuracy by approximately 1%. The layer was tested without padding, i.e. ignoring the word indices in the margins (start and end of the text) when performing convolutions. The alternative of adding 0-based word indices in the margins gave results of the same significance in accuracy. This may be explained by the large quantity of filters used and by the fact that the mask size is 3x3, which makes the margins barely representative when convolving the 1000 word embeddings. Masks of higher size (e.g. 5x5 and 7x7) were also tested, but they yielded less accuracy (between 1% and 2% less), since the model can be more precise when observing more local links between words, rather than long sequences, more likely to span through different sentences.
4. Global maximum pooling layer: selects the word embedding with maximum value for each filter of the previous layer. This acts as a summary of the evaluation of each convolution filter for the text, directly useful for the classification task.
5. Single-output-unit dense layer: uses 500 hidden units to learn the effect of the 300 in the classification task, updating the weights with backpropagation. The selected number of hidden units was slightly better than lower alternatives (e.g. around 0.5% better than with 300), and not significantly less accurate than larger options. The activation function that determines the final result is the sigmoid function, especially suited for binary classification scenarios like this one due to its two-regions-based shape.

Since this is a binary classification problem, binary cross-entropy is a natural choice as a loss function. Different optimizer techniques (related with the model's learning rate) have been tested (Nadam, Adam, Adadelta), yielding similar results, i.e. without any of them being significantly better than any other, which may be caused by the nature of the problem, lacking the easy-to-fall-in local minima present in some other cases, such as some regression problems.

Different batch sizes were tested, and 25 was selected since it was the one yielding faster convergence of the model, without decreasing the accuracy compared to others (e.g. 32 or 50). The model achieves it maximum validation accuracy in the 2^{nd} epoch, with larger number of epochs causing overfitting: the validation accuracy

decreases even though the training value becomes higher. This phenomenon is represented in Figure 2. After learning and validation, the resulting model is evaluated on the test set. The average accuracy of 10 executions is 0.899, and the loss is 0.214. The accuracy result is slightly higher than 0.89, which is the value reported by other sources that have solved the same classification problem for IMDB using CNNs, such as [2] and [5].

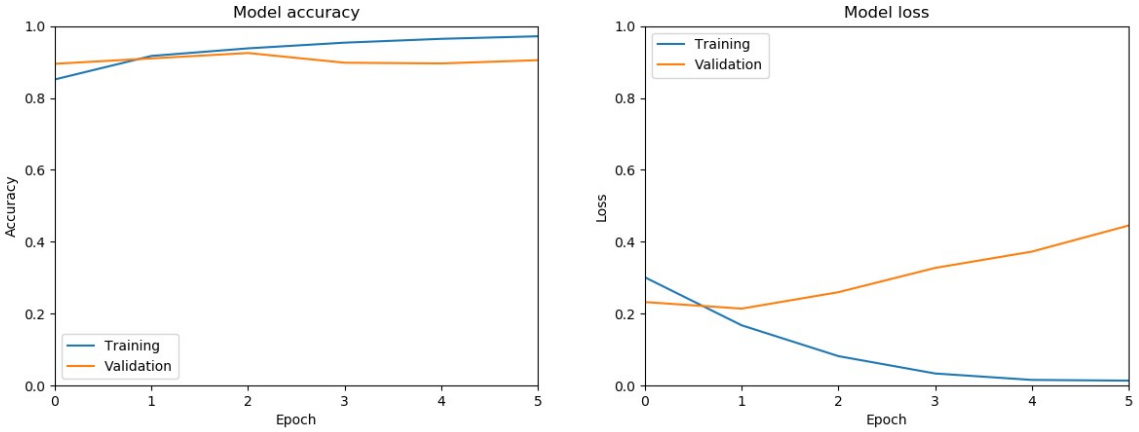

Figure 2: Training and validation functions for the basic model's accuracy (left) and loss (right). It can be seen that overfitting occurs after the 2nd epoch (labeled as 1).

3.12. Refined architecture

The model complexity is increased in this architecture with the will to learn more complex patterns in the review's text and therefore increase the accuracy. For this reason, an additional dense layer with 500 units is added between the max pooling operation and the dense layer that produces the output. In this case, the activation function is ReLU, since it has been reported to work well for a large variety of problem types.

The basic architecture reaches the overfitting point very quickly, and from then onwards it stops generalizing. The addition of an extra dense layer to increase the model's complexity may contribute to make the overfitting faster and more severe if no generalization technique is used too. To increase the generalization capacity of the model, regularization has been introduced. In particular, dropout layers are added to the model to ignore neurons with a 30% probability at each pass of the learning process. This is applied in two points of the architecture: firstly, between the embedding and convolution layers, to increase the generalization of word embeddings and get more descriptive filter patterns; secondly, between the newly-introduced dense layer and the final dense layer, to avoid that the new layer learns intricate, training-exclusive patterns. The resulting model is shown in Figure 3.

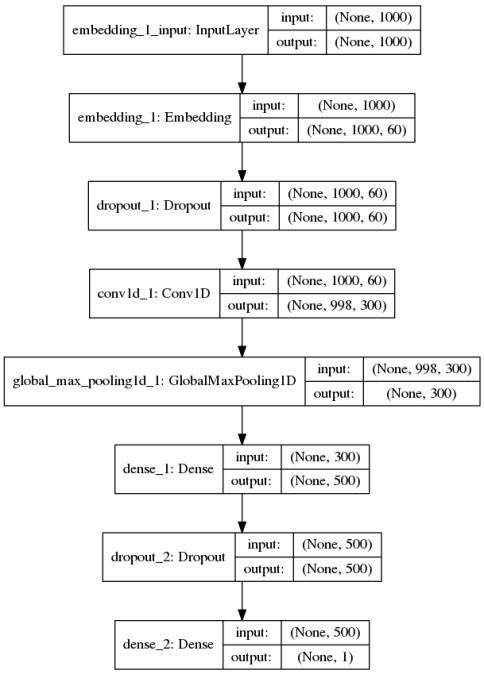

Figure 3: Final model architecture.

The refined model successfully delays overfitting to the 3rd epoch, that is the point where the validation accuracy is maximum, and therefore the final model should be limited to 3 epochs. The accuracy is also improved with the addition of the new dense and dropout layers, reaching 0.916 accuracy and a loss of 0.211 as an average of 10 experiments. The result is shown in Figure 4.

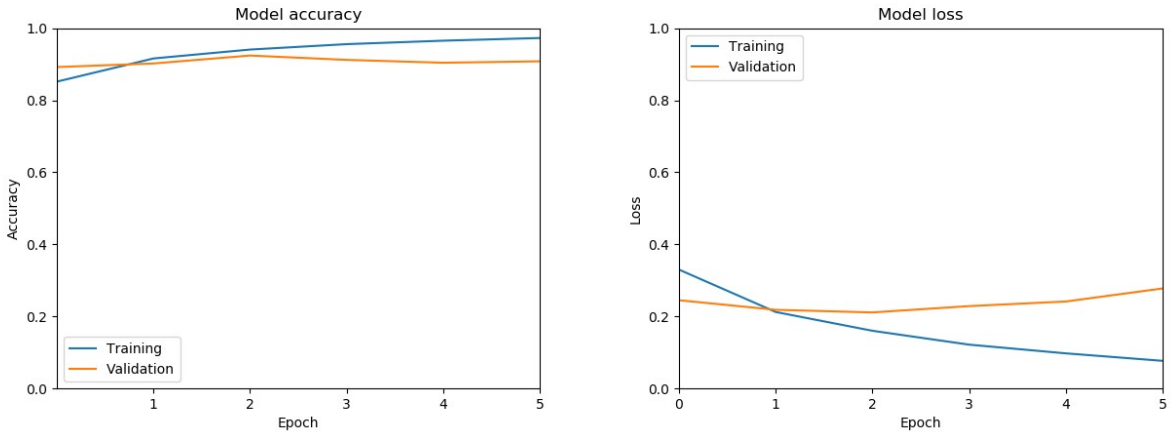

Figure 4: Training and validation functions for the refined model's accuracy (left) and loss (right). It can be seen that overfitting occurs after the 3nd epoch (labeled as 2).

As depicted by the confusion matrix of Figure 5, the model is almost equally precise in determining that positive reviews are actually positive as it is to determine that negative ones are indeed negative. This is not a surprise, since a same proportion of both classes was used for training (and also for validation). However, the times that a negative review is considered positive is considerably higher than the other way round, which may be caused by the used of sarcasm or irony in negative reviews (using words and phrases that are also present in positive ones), that cannot be identified without deeper context analysis of the text, or external word knowledge. The false positives and true negatives, nevertheless, are more than 10 times less frequent than the true positives (84 in contrast to 916), as previously shown by the accuracy results.

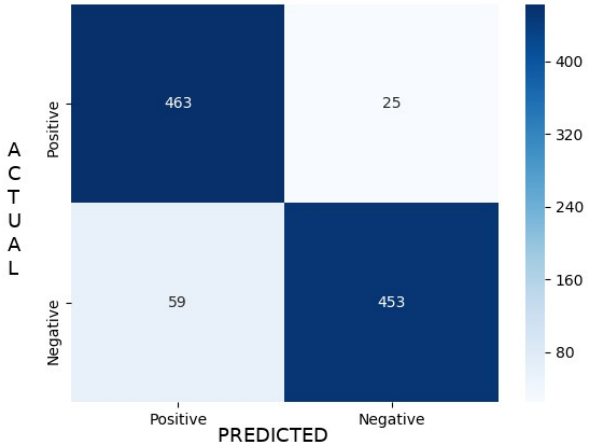

Figure 5: Confusion matrix of the model, showing the predictions of each class of review (positive and negative).

4. Conclusions

The results of the basic architecture are good for the context of the problem, but the state of the art for IMDB is higher [4], with 0.945 accuracy. The state-of-the-art paper describes the usage of techniques such as paragraph vectors and long-short term memory (LSTM) that were not used in this work, but, nevertheless, the refined CNN-model implemented here achieves 0.916 test accuracy, a result that is clearly higher than other CNN approaches ([2], [3] and [5]). This work may be improved with the inclusion of the cited state-of-the-art techniques (paragraph vectors and LSTM) to exploit the sequential order of the words in the text to gather more information for the predictions of the model.

5. References

Developed code:
[1] Github repository

Checked sources:
[2] Brownlee, Jason (Machine Learning Mastery), 2017. *How to Use Word Embedding Layers for Deep Learning with Keras*.

[3] El Boukkouri, Hicham (Medium), 2018. Text Classification: The First Step Towards NLP Mastery.

[4] Hong, J., & Fang, M. (2015). Sentiment analysis with deeply learned distributed representations of variable length texts. *Stanford University Report*.

[5] Keras Team, 2019. *Example of IMDB CNN*.

[6] Rajat (Towards Data Science), 2018. *Sentence Classification using CNN with Deep Learning Studio*.