Yu Lu Liu

Final Project MAIS 202 Deliverable 2

Problem Statement

Identifying whether a written statement is sarcastic is one of the challenges in sentiment analysis. This project aims to detect whether a news headline expresses sarcasm or not.

Data Preprocessing

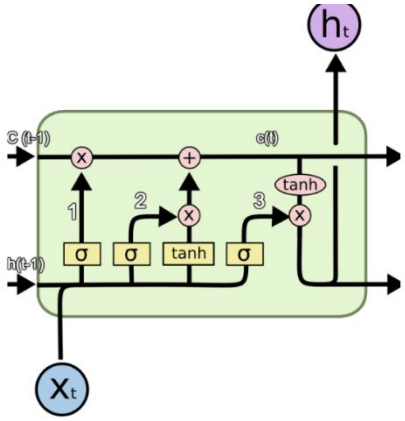
The main dataset chose is the News Headlines Dataset for Sarcasm Detection from R. Misra and A. Prahal (2019). This dataset is originally organized into three columns. The first one contains the links to the articles, which are not needed for this project. I therefore deleted it. The remaining columns contains the headlines and a binary indicating whether the corresponding news article is sarcastic (1) or not (0). There's 14985 non-sarcastic headlines and 11724 sarcastic ones.

To transform each headline into a vector, I used the Tokenizer from Keras' text preprocessing library. It lowercased the text, deleted punctuation, and created a dictionary in which each word is associated to a number. I decided to conserve only the 5000 most common words to make the algorithm easier to run. So, the Tokenizer transformed each headline into a vector which contains the numbers (1-4999) associated to the original word. I then padded them, so each vector is of equal size.

I then split the data into three: training set (70%), validation set(15%) and test set(15%).

Machine Learning Model

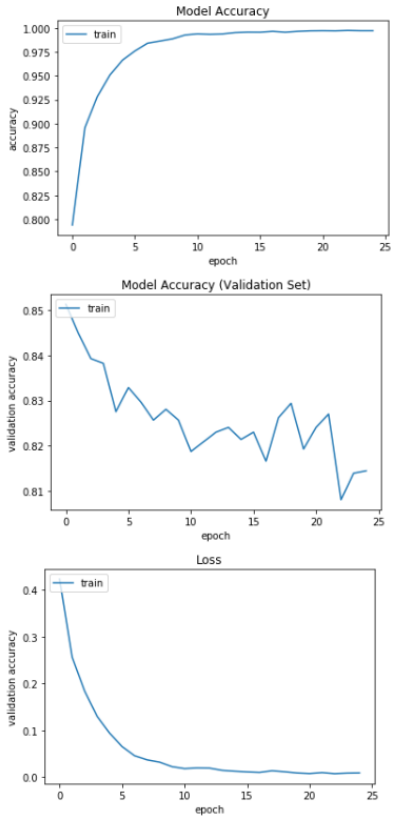
The machine learning model I chose is Long Short-Term Memory Network (LSTM), which can be considered as an improvement of Recurrent Neural Network (RNN) in terms of memory. In fact, unlike the RNN, which can only conserve the output of the immediate previous step, LSTM contains gates (sigmoid layers) that can be trained to remember or to forget. Therefore, if a relevant information is presented at the start, given that the gates are trained properly, LSTM has access to it in subsequent steps. This ability to remember is needed in this project since context is important in order to determine whether a statement is sarcastic or not. If RNN is used instead, it is very probable to forget important information from previous steps.



A LSTM unit. (Source : http://colah.github.io/posts/2015-08-Understanding-LSTMs)

To implement, I used Keras. The first layer is an embedding layer which encodes the vocabulary words in relation to each other in a vector space. The only hyperparameters involved in this layer is the embedding dimension and the dropout rate, which I set to 128 and 0.2, standard values for a large dataset like the one I used. The second layer is the LSTM itself, which also includes dropout layers to reduce chances of overfitting. Finally, the last layer is an activation layer, which is a softmax activation function that outputs probabilities. The loss function is a binary cross-entropy function since the model is doing a binary classification. The model will be judged based on its accuracy, and the optimizer, adaptive moment estimation, is also chosen because it is a popular and practical optimizer.

Preliminary Results



To fit the model to the training set, it is iterated 25 times. 20% of the training set is set aside as the validation set to measure the model's accuracy at that time.

As shown in these graphs, the model's accuracy as measured to the training set has increased, stabilizing after 10 epochs. The loss with respect to the training set has also decreased accordingly. However, the validation set's accuracy has actually decreased, which leads me to believe that the model overfitted to the training set.

When the model is made to predict the test set, it achieved an accuracy of around 81.4%, a precision of 83.2% and a recall of 72.4%. It is clear that there's ample room for improvement and that the model, by overfitting to the training set after the 10th epoch, was unable to learn more.

Next Steps

Since the problem seems to be overfitting, the next steps would be to address this issue. I could explore different forms of embedding to reduce the embedding space and thus reduce chances of overfitting. I could also tune the hyperparameters of the embedding layer such as increasing the dropout rate. Once the problem of overfitting addressed, I would like to improve the model's accuracy by optimizing the hyperparameters of the LSTM layer and of the dense layer.

References

Sinha Nimesh. 2018. Understanding LSTM and its Quick Implementation in Keras for Sentiment Analysis. (https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47).

Rishabh Misra, Prahal Arora. 2019. Sarcasm Detection using Hybrid Neural Network. (Kaggle dataset: https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection/metadata).