Syntactic and Semantic influence in deep learning models for Text Classification tasks

Assa, Yuval
ID: 318612025

Daniel, Sean
ID: 315619643

Submitted as final project report for the NLP course, COLMAN, 2021

1 Introduction

We chose our topic because we were interested in investigating the recent popular topic - interpretation of deep neural network models. In fact, understanding rather a high accuracy neural network takes in consideration the syntactic and semantic factors of text in order to classify a given sentence correctly. In other words, are there any keywords / important n-grams that are excessively influential in a sentence - more than the Syntactic / Semantics of a sentence in terms of classifying tasks.
We believe that the results of this experiment can improve our later NLP tasks such as our final NLP project.
We searched for some papers regarding this problem - they will be discussed later in this paper.

1.1 Related Works

1. ShufText: A Simple Black Box Approach to Evaluate the Fragility of Text Classification Models.
2. Deep text classification can be fooled.
3. Benchmarking popular classification models' robustness to random and targeted corruptions
4. AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models.
5. Understanding neural networks through representation erasure.

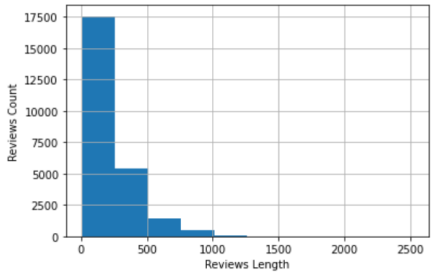
2 Solution

2.1 General approach

The approach is training a neural network with a given movie reviews dataset while tracking the loss, validate and tune the model's hyper-parameters using validation set, test and record the accuracy of the test set, then check if the neural network is ignoring the Syntactic / Semantics of the sentences - we do so by creating a shuffled test set where the words in the test sentences are shuffled randomly but the labels still correspond to the original sentences and test the model again using the shuffled test set.
High accuracy on this test set would imply that the model has not learned the sentence structure or semantics and is relying on the key-words.
Another experiment to test rather a neural network is highly relient on key-words and not on the Syntactic / Semantics of a sentence is by training the neural network on shuffled train data. At last, compare the accuracy between the two tests in the two experiments. We decided to use a Sentiment Treebank that is classified in a binary way as our dataset. It contains 25,000 movie reviews, with different review length. Each review is classified as positive or negative. In overall, the dataset has 74,072 unique words.

The reviews length (measured by number of words) distribution is as follows:

2.1.1 Reviews Length Distribution




```
count    25000.00000
mean       240.80784
std        179.01773
min         10.00000
25%        130.00000
50%        179.00000
75%        293.00000
max       2514.00000
dtype: float64
```

2.2 Design

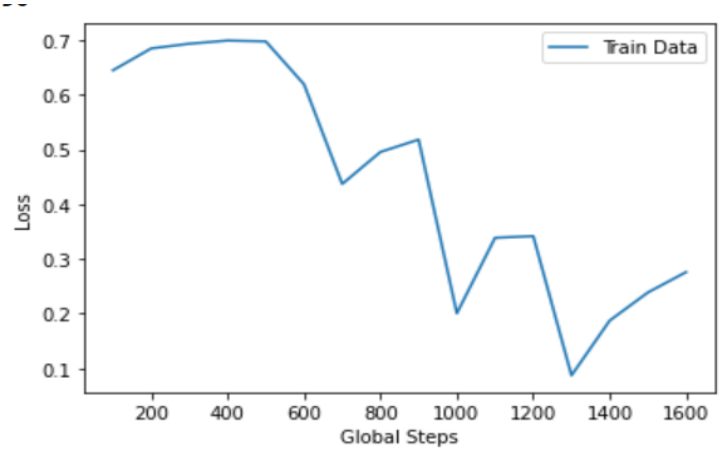
We wrote our code in Jupyter notebook on Google Colab platform.
After trying different model like BERT ("bert-base-uncased" - with 12-layer, 768-hidden, 12-heads) which was pre-trained tokenizer and its corresponding

word-to-index mapping, we found out that based on our data that we provided the model didn't do so well.
So we decided to define our own model - embeddings and a bidirectional, word-level, 2 layer LSTM network as the classifier with a dropout layer with probability of 0.3 and a linear layer from "hidden_dim" (256) to "output_size" (1) and a Sigmoid layer stacked on top.
First, we loaded the movie reviews and their labels (positive/negative) from the data folder. Then we removed all the unnecessary punctuation from the reviews. We started by creating our own vocabulary. We did so by converting words to integers based on the number of occurrence of the word. Then we created a dictionary to convert words to Integers based on the number of occurrence of the word in the reviews. At last, We encoded the review in to list of Integer by using the dictionary.
We set up our neural network model and trained the net with the training data. The training process took 1min and 46s for 4 epochs. While training the net, we validated it with the validation dataset and tuned the net's hyperparameters according to the results - in order to preform better. After all, we ran the test dataset through the net and record the accuracy. We did the same process for the two experiments mentioned in the next section.

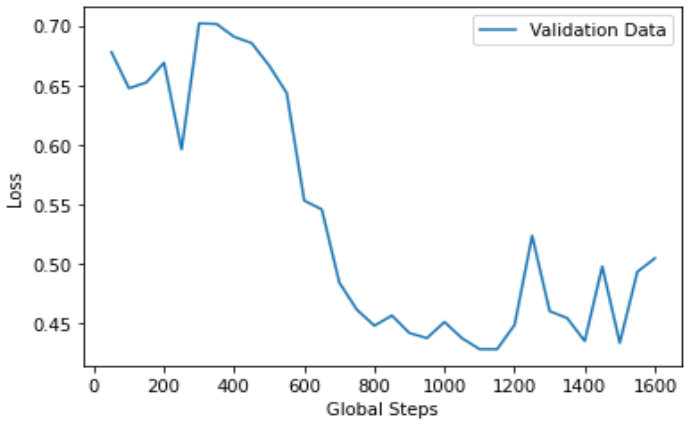
3 Experimental results

As mentioned above, we used embeddings and a bidirectional, word-level, 2 layer LSTM network as the classifier with a dropout layer with probability of 0.3 and a linear layer from "hidden_dim" (256) to "output_size" (1) and a Sigmoid layer.
To tune our net, we had to do some experiments with the net's hyperparameters. We found out that our net got the highest accuracy by training the net with batches of 50 reviews at a time, dropout inside the LSTM with probability of 0.5 and learning rate of 0.001.
We used BCELoss as criterion and Adam as optimizer.
We divided our rating dataset (total of 25,000 movie reviews) into train, validation and test sets with the percentage of 80%, 10%, 10% accordingly.
We trained our net with different epoch numbers to see what is the ideal number of epochs.
Below is a comparison between different epoch numbers and how the loss curve was effected (both train and validation data) as well as the test accuracy:

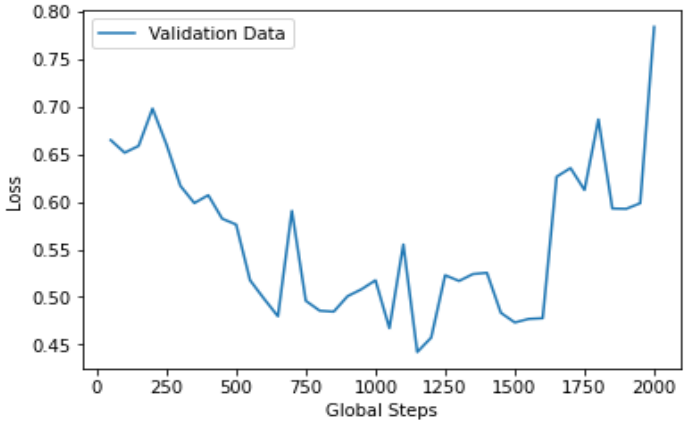
3.0.1 Training loss in each step - 4 epochs



3.0.2 Validation loss in each step - 4 epochs



3.0.3 Validation loss in each step - 5 epochs



3.0.4 Overall Epochs test

# Epochs	Train Loss	Validation Loss	Accuracy
1	0.525565	0.606192	0.683
4	0.246203	0.504509	0.815
5	0.163233	0.783522	0.761

-For only one epoch, the net is under-fitted, the accuracy is low.
-We can see that the best results were received by training the net for 4 epochs.
-For 5 epochs we can clearly see how the net is tending to be over-fitted (by the sharp increase of the validation data loss curve).

As mentioned above, we took in consideration two different experiments. The first experiment was done by testing the trained net with randomly shuffled reviews from the test dataset and compare between the accuracy.
The experiment results are as follows:

3.1 First experiment results

Test Data Type	Loss	Accuracy
Regular Data	0.485	0.815
Randomly Shuffled Data	0.515	0.798

The second experiment was done by training the new with randomly shuffled reviews from the training set and testing it with normal non randomly shuffled reviews.
The experiment results are as follows:

3.2 Second experiment results

Train Data Type	Loss	Accuracy
Regular Data	0.426	0.818
Randomly Shuffled Data	0.48	0.804

4 Discussion

After conducting the experiments mentioned in this paper, we can conclude that deep neural network models are heavily reliant on important keywords or n-grams relevant to the classification task and tend to completely ignore the Syntactic / Semantics understanding of a sentence.
We can also say that it is very hard to interpret a deep neural network, it is usually preform like "black box".

5 Code

https://github.com/sean1515/NLP_FINAL_PROJECT

References

[1] ShufText: A Simple Black Box Approach to Evaluate the Fragility of Text Classification Models,
https://arxiv.org/pdf/2102.00238.pdf

[2] Gao, J., Lanchantin, J., Soffa, M.L., Qi, Y.: Black-box generation of adversarial text sequences to evade deep learning classifiers. CoRR abs/1801.04354 (2018),
http://arxiv.org/abs/1801.04354

[3] Desai, U., Tamilselvam, S., Kaur, J., Mani, S., Khare, S.: Benchmarking popular classification models' robustness to random and targeted corruptions (2020),
https://arxiv.org/pdf/2002.00754.pdf

[4] Deep text classification can be fooled,
https://arxiv.org/ftp/arxiv/papers/1704/1704.08006.pdf

[5] AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models,
https://arxiv.org/pdf/1909.09251.pdf

[6] Understanding neural networks through representation erasure,
https://arxiv.org/pdf/1612.08220.pdf

[7] Google Colab Platform,
https://colab.research.google.com/