Final Project Report for CS 175, Winter 2020

Project Title: Movie Review Classification

Student Name(s)
Darren Lim, 24233004, limdj1@uci.edu
Jerson Villanueva, 77425248, jersonv@uci.edu
Justin Lonh, 65326458, jlonh@uci.edu

1. Project Summary

Our project is about classifying movie reviews using sentiment analysis using Python. We used data sets found online, such as one data set which includes processed movie reviews from kaggle.com made by Bo Pang and Lillian Lee, and feed them through a neural network to predict whether reviews are positive or negative. Our goal is to compare different neural networks (convolutional neural network and recurrent neural network) and their performance on these data sets. We also want to see how these techniques compare to a pre-trained model (GloVe with neural network), as well as see how different amounts of data affect the accuracy of the neural network. We concluded through training and testing that the convolutional and recurrent neural network have consistent and high accuracies, around 75 - 80% while the GloVe model did not have as high accuracy. We used the RNN model to test on the other data training set (ex. RNN Kaggle trained model tested on IMDB test set) and the accuracies were around 50%.

2. Data Sets

We have found two data sets that are available publicly; these two data sets contain both positive and negative movie reviews as well as labels.

The first data set was created by Bo Pang and Lillian Lee, which we found on Kaggle, so we will refer to this data set as the Kaggle data set. It contains a total of 2000 reviews (documents) in which 1000 reviews are labeled as positive and the other 1000 are labeled as negative. Conveniently, the reviews are already processed, which means we only need to deal with punctuation and splitting by words. We are splitting the documents by 90% train and 10% test. Of the 90% training documents (about 1800 reviews), we clean out stop words, non-english words if included, punctuation, and words that are smaller than a character, and finally stem the words to get our total vocabulary: 29581 words. From there, we remove words that occur less than 2 times, because we find words that do not occur many times are useless in the training process. The final vocabulary we are left with is about 17576 words. Here is a sample of a positive review from the data set:

```
the rest of the story isn't important because all it does is serve as a mere backdrop for the two stars to share the screen .
sure , there are some mildly interesting subplots , but they all fail in comparison to the utter cuteness of the main relationship .
all of this , of course , leads up to the predictable climax .
but as foreseeable as the ending is , it's so damn cute and well-done that i doubt any movie in the entire year contains a scene the
evokes as much pure joy as this part does .
when ryan discovers the true identity of her online love , i was filled with such , for lack of a better word , happiness that for the
first time all year , i actually left the theater smiling .
```

https://www.kaggle.com/nltkdata/movie-review

The other data set was created by Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher. These movie reviews are from IMDB, so we will refer to this set as the IMDB data set. In this data set, there are 25,000 reviews designated for training, 25,000 designated for testing, as well as additional unlabeled reviews that can be used. This data set also provides a raw text, and a bag of words that has already been processed that we can use as well. However, we cleaned this data set ourselves and created our own vocabulary with it. We also only used the 25,000 training and 25,000 testing reviews for this project. The total amount of words we cleaned from this data set was 89,547 words. We also removed words that occur less than 2 times in the vocabulary. The final vocabulary we got from this data set is 39,515 words. Here is a sample from the negative reviews:

```
he has failed in it very badly. I guess he was trying to make a
stylish movie. Any way I think this movie is a total waste of
time and effort. In the credit of the director, he knows the
media that he is working with, what I am trying to say is I have
seen worst movies than this. Here at least the director knows to
maintain the continuity in the movie. And the actors also have
given a decent performance.
```

https://ai.stanford.edu/~amaas/data/sentiment/

With these two separate movie review data sets cleaned and ready to use, we then split the reviews into training and testing sets. Within those sets, we changed the words into sequences of integers and padded those sequences in order for each review to be the same length. We padded them by finding the longest review from each training and testing set, and set each review sequence to have the same length, appending 0s if the length of the review is too short.

```
max_length = max([len(s.split()) for s in train_docs])
Xtrain = pad_sequences(encoded_docs_train, maxlen=max_length, padding='post')
```

```
[   50    49    39   236   209    77   896    50     7    13  1647    74
    26  1858   928   267   387   304   601     3    14    16     1  4359
    74    20     1    54     1    31     8     1  1858   258     1     6
   314  4948   270    63    17    84  2341     1     4    17    20    40
    36     4   290     7   845   544     4    75    26  2633  2335    12
    64   110     3     1     2    55   115   748   293   260     3     9
    64  3737     3    24   296    27   577     6   175     4     6     3
   313  6588  5233   265    57    11   680   220     2     1    59    20
   140    65    30   628     1     1    92     4     3     9   229     7
   255  1714  1471  1335    35     9    12    30     8   111  2459  5718
    57   102     1   393    40    40    40     1   175   600   920  1050
   911    15    17  1027  2275    42   101    61    14     3  3020     1
    63   153  1746     3   126     4    81    84    27   989    33    37
    31    25     7    32  3123    15    19     3  2399    24    68    31
     3    48   241     2   318     3  1713     8     3   177    16   103
     1     2    17   318 10238    37  1008   199     2   157   106     2
   500     3    29    80     3    29    91    22   374   106     7   357
    85     3   628   834     2   342    13     3     1  4948     3   241
   342     3    75  3221  1092     1  1662    85  6617     1     9   275
   577   549   577    19    12     3    59    50    74   561     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

At this point, the reviews are now ready for training and testing.

3. Technical Approach

To classify movie reviews into positive and negative reviews, we trained a convolutional neural network (CNN), a neural network with a GloVe pre-trained embedding model, and a long short term memory recurrent neural network (LSTM RNN). We evaluated and compared the differences between each model by how well they did in testing. Some of the technologies we use for building the vocabulary, network, and evaluation includes tensorflow, keras, numpy, gensim, and nltk. We have decided to use the convolutional neural network because we have found through our research it is one of the standard neural networks to train document classification on. (https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/ , https://towardsdatascience.com/cnn-sentiment-analysis-9b1771e7cdd6) We use the neural network with GloVe embedding model because we wanted to have a model that included a pre-trained embedding to compare. Finally, we use a recurrent neural network model through the suggestions of the TA. We used to have two other neural network models: a neural network with one-hot encoding and another neural network with a word2vec pre-trained embedding model. However, we decided to remove these two models and focus on the pre-trained embedding that did the best, which was the GloVe embedding model.

The first process is to decompose the movie reviews into a vocabulary we can use for the word embeddings. To prepare the data, we first clean the data sets by removing unnecessary punctuation, words that are not english, stop words, and finally stemming.

For the Kaggle movie reviews, we used 90% of the documents as training data and the remaining 10% as testing data (1800 reviews for training, 200 reviews for testing). For the IMDB data set, we used 50% training data and 50% testing data (25,000 reviews for training, 25,000 reviews for testing). Afterwards, we created our own vocabulary from the cleaned reviews and converted the reviews to sequences of integers then padded them, as described in the data sets section above. From here, we created a CNN model, a neural network with GloVe embedding model, and a LSTM RNN model.

For the first model of training on our own word embedding, the neural network is defined as such: Embedding layer as the first layer, then a CNN, a pooling layer, then a Multilayer Perceptron layer, and an output layer using a sigmoid function.

GloVe is a pre-trained word embedding, so all we needed to do to create a model using GloVe was to download the text file that contained all of the word embedding information, and then load it into our notebook. After that we just needed to go through our vocabulary, and then get the vector for each word from the text file, and put them into a matrix. Lastly, we use this matrix as the weights for our embedding layer.

Using a pre-trained model is very simple, we just had to load the embeddings from a text file, and put the information that was there into a weight matrix that we can use. The data pre-processing step is the same in our first model. After we do that, we just create the neural network with the same layers as the previous models, fit the training data, and then evaluate its performance on the test data.

For our last model, the LSTM RNN model, we created an embedding layer, then used bidirectional layers, and lastly an output layer using relu.

Please refer to appendix B for each of the models' summaries / layers.

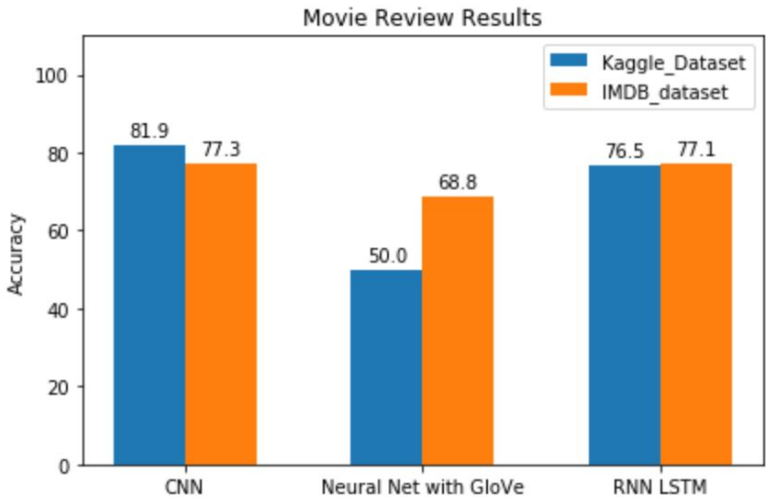
We used accuracy percentage to evaluate our models. The accuracy is important to our evaluation functions because it is how we compare and contrast the different neural network models as well as the models using different amounts of data. To calculate the accuracy, we tested the trained models on a predetermined amount of testing data that has already been sorted positive or negative. The accuracy is the overall score of whether the network correctly determined the review to be positive or negative.

```
Test Accuracy CNN IMDB: 80.98%
```

4. Experiments and Evaluation

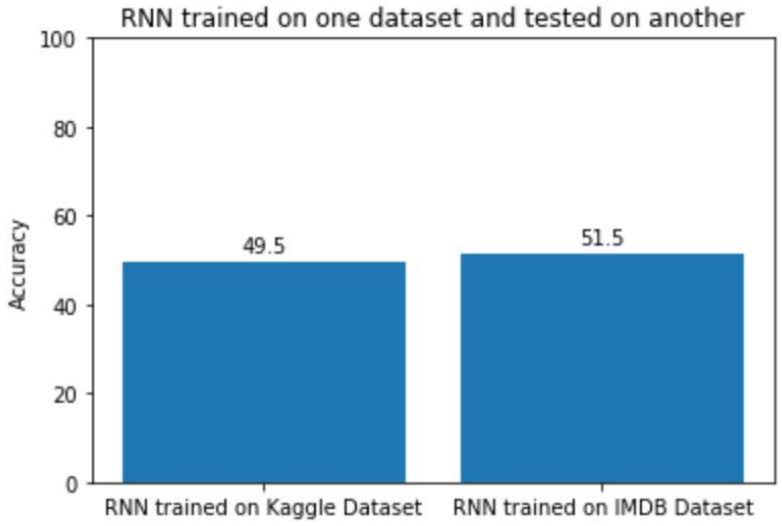
In the first phase of our project, we conducted a few preliminary experiments with different models and techniques. We built a neural network model and incorporated different techniques such as bag of words, one-hot encoding, and word2vec to get a general idea of how well these techniques worked compared to each other. In our initial results, we found that a bag of words, one-hot encoding, and word2vec did not have good results; these two techniques ended up having around only 50% accuracy on test data, which was significantly lower than the accuracies we found for the other experiments. For example, our preliminary experiments gave GloVe an accuracy of about 70%. The GloVe model was also much faster than one-hot encoding and the bag of words models. These results were also consistent with what we expected at this point, as from previous lectures and research we already knew that bag of words, one-hot encoding, and word2vec were some of the simpler techniques used for natural language processing. At this point, we decided to drop bag of words, one-hot encoding, and word2vec from our experiments and continue focusing more on experimenting with the other techniques that showed more promise, which included convolutional neural network, GloVe with neural network, and recurrent neural network.

From there, we transitioned to experimenting to determine the differences in effectiveness of a convolutional neural network, GloVe with regular neural network, and recurrent neural network on our movie review data. We trained six models total (three models based on different techniques for each of the two datasets). Using the same 50/50 split of training/testing data, we evaluated the accuracies of the models for each dataset. At the end, as a small additional experiment, we also evaluated the accuracy of the RNN models on the testing data from the dataset that it was not trained on (i.e. evaluating the RNN trained on the Kaggle training data with the IMDB testing data, and evaluating the RNN trained on the IMDB training data with the Kaggle testing data).



Based on our experiments and evaluations of accuracy, we found that the neural network with GloVe performed the worst out of the three models we used at around 50% accuracy for the Kaggle dataset and 68.8% accuracy for the IMDB dataset. For the CNN model, we got an accuracy of 81.9% for the Kaggle dataset and 77.3 % for the IMDB dataset. Then for the RNN, we got 76.5% for the Kaggle dataset and 77.1% for the IMDB dataset. These results are shown in the graph above.

The difference in the accuracies for the two datasets was very miniscule for the RNN model with a difference of .4%. The difference was greater for the CNN model with a difference of 4.6%, and it was the greatest for the GloVe model with a difference of 18.8%. The results for these two datasets were supposed to represent what effect the size of the dataset has on the accuracy of the model, but it was brought to our attention that there could be other differences in the datasets aside from the amount of data, like the length of the reviews, which could also affect the accuracy of the model. So, the results we received might not show how the size of a dataset actually affects the accuracy of a model.



We used our RNN model when we experimented with training the model on one dataset and then testing it on the other dataset. This is similar to how the model would be used in the real world. The model would be trained, and then used to classify reviews that may or may not be similar to the reviews that it was trained on. This of course would greatly affect the accuracy of the model, and that is shown in the results. In both scenarios, the accuracy was around 50% which is the accuracy one would get if they were to classify the reviews randomly. This shows that it is necessary for our model to be trained on the data that is similar to the test data to be able to perform well.

Another interesting experiment we could have done if there was enough time would be to train the model on both datasets and then test on the test data from both datasets separately and see what effect it has on the accuracy of the model. This would show us how training the model on more data outside of the dataset would affect the accuracy of the model on that dataset.

5. Lessons Learned and Insights

One insight we have gained through this project is that having a larger data set does not necessarily have the biggest impact on performance. Looking at the CNN and RNN test results, the models had similar accuracy results for both datasets. The only exception is with the GloVe model where the IMDB set did better than the Kaggle set. As such, it is possible that data set size may have different impacts on overall accuracy depending on the technique used to create the model. In the future, further research on this idea could be done by using a single dataset and testing how the accuracy of each model changes given different amounts of training data. For example, training each model on subsets of 1000, 5000, 10000, 15000, and 20000 training data points.

Another insight we gained is we thought using pre-trained embeddings such as the GloVe model was going to do as good if not better than using a CNN or RNN. However, we found that the GloVe model and other pre-trained embeddings we used during the initial experimentation trials, one_hot encoding and word2vec, did worse than our CNN and RNN. As this did not meet our expectations, we were surprised to find that GloVe did not do as well. We did expect the CNN and RNN to do decently well, and the results of our experiments met our expectations.

Another insight that we gained is from testing our RNN model on one dataset, and then testing it on another dataset. This is similar to what it would be like in the real world if a model was being used to classify movie reviews. The model will not be able to train on the reviews before it tries to classify them. The resulting accuracy was about 50% accuracy which is pretty much the same as classifying the reviews randomly. So, in its current state, our model can only do about as well as if we were to classify the reviews as positive or negative at random. Our model in its current form is far from being able to replace humans in classifying reviews.

APPENDICES

Appendix A*:* Software

MovieReview.ipynb - our Jupyter notebook used to code and assemble the different parts of the project and conduct our experiments

We looked at the websites below to help us implement our models.

https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/

https://www.tensorflow.org/tutorials/keras/text_classification_with_hub

https://machinelearningmastery.com/predict-sentiment-movie-reviews-using-deep-learning/

https://jjallaire.github.io/deep-learning-with-r-notebooks/notebooks/3.4-classifying-movie-reviews.nb.html

https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

https://www.tensorflow.org/tutorials/text/text_classification_rnn

https://towardsdatascience.com/cnn-sentiment-analysis-9b1771e7cdd6

Appendix B: Additional Results

CNN Layers

Layer (type)	Output Shape	Param #
embedding_7 (Embedding)	(None, 942, 100)	1060800
conv1d_7 (Conv1D)	(None, 935, 32)	25632
max_pooling1d_7 (MaxPooling1	(None, 311, 32)	0
flatten_7 (Flatten)	(None, 9952)	0
dense_7 (Dense)	(None, 15)	149295
dense_8 (Dense)	(None, 1)	16

Total params: 1,235,743
Trainable params: 1,235,743
Non-trainable params: 0

GloVe with Neural Network Layers

Layer (type)	Output Shape	Param #
embedding_9 (Embedding)	(None, 942, 100)	1060800
flatten_9 (Flatten)	(None, 94200)	0
dense_11 (Dense)	(None, 15)	1413015
dense_12 (Dense)	(None, 1)	16

Total params: 2,473,831
Trainable params: 1,413,031
Non-trainable params: 1,060,800

LSTM RNN Layers

Layer (type)	Output Shape	Param #
embedding_12 (Embedding)	(None, 942, 100)	1681700
bidirectional (Bidirectional	(None, 942, 128)	84480
bidirectional_1 (Bidirection	(None, 64)	41216
dense_17 (Dense)	(None, 64)	4160
dropout (Dropout)	(None, 64)	0
dense_18 (Dense)	(None, 1)	65

Total params: 1,811,621
Trainable params: 1,811,621
Non-trainable params: 0