SentiNel

Final Project Report

ECE324

Sonali Dey (1004389627)
Ayushi Shrivastava (1004146033)

Prof. Jonathan Rose

Division of Engineering Science
University of Toronto

Word Count: 1871
Word Count Penalty: 0%

3 December 2019

1 Introduction - Project Goal and Description

Neural networks have been shown to be applied successfully to Natural Language Processing applications [1]. With this in mind, SentiNel attempts to detect the positive, negative, or neutral sentiment associated with the each 'named entity' i.e. noun in sentences on Twitter and in film reviews using neural networks.

For instance, in the sentence, "The plot was gripping, but the special effects were lacking", SentiNel should identify the positive sentiment expressed towards 'plot' and the negative sentiment expressed towards 'effects'.

2 Illustration

The following is an overview of the originally intended functioning of SentiNel:

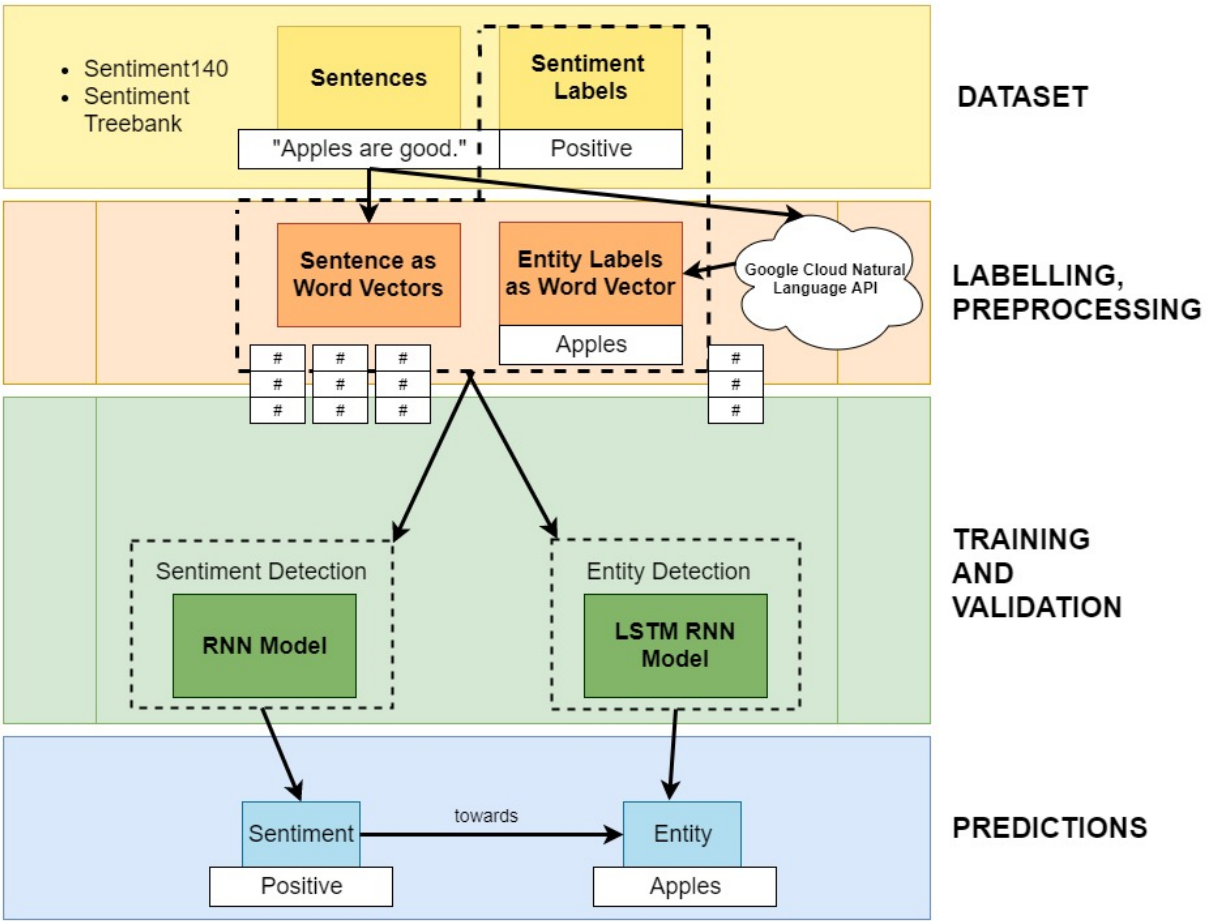

Figure 1: General Project Overview

Due to the challenges faced during entity analysis, the updated structure follows:

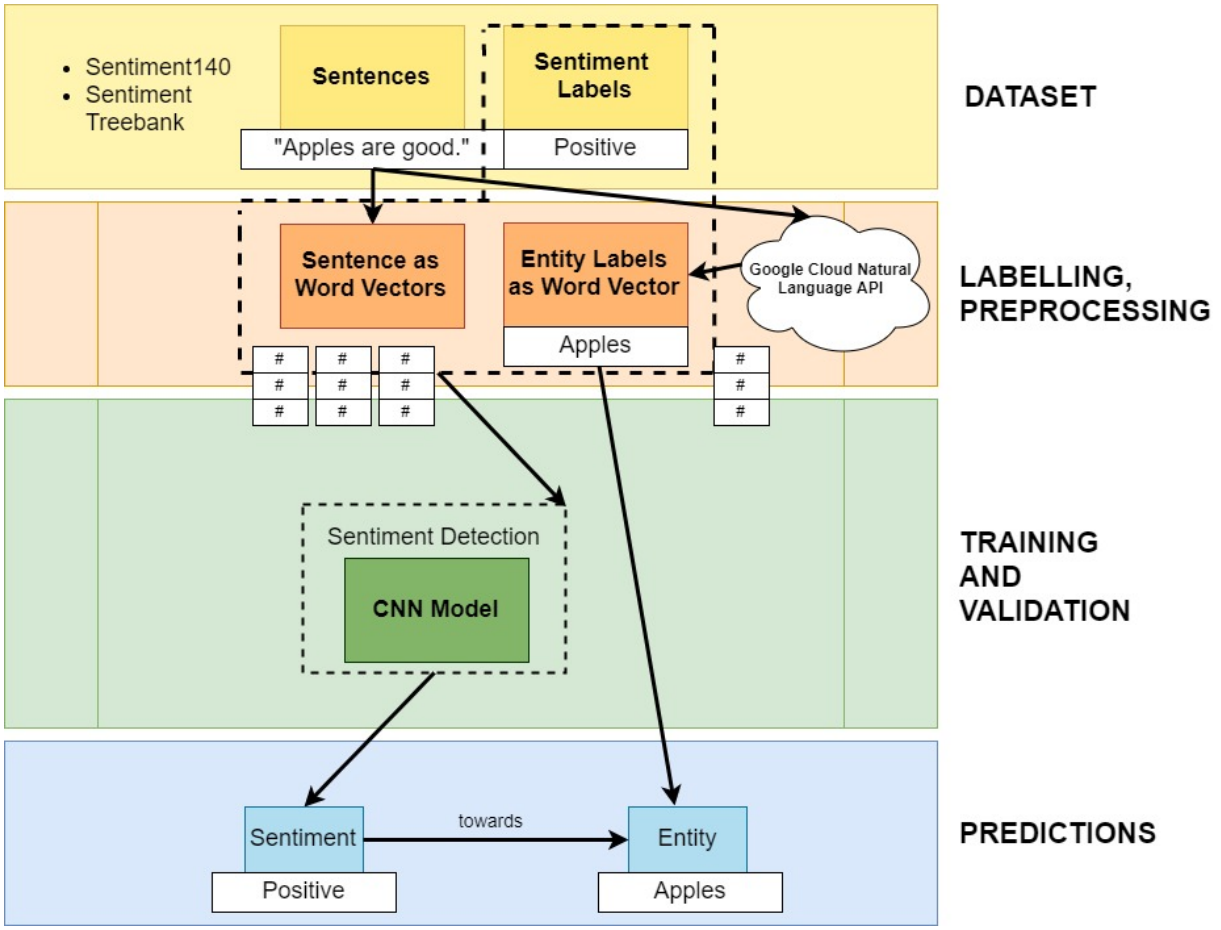

Figure 2: General Project Overview

3 Background and Related Work

Large amounts of research has been conducted for Natural Language Processing applications including sentiment analysis. For instance, using the Sentiment Treebank dataset, a research group at Stanford University predicted positive or negative emotions in a sentence with 85.4% accuracy using their Recurrent Neural Network [2].

For named entity recognition and associated sentiment analysis, several approaches have been made in research papers that contributed to the ideas for our model. For instance, 85%+ precision, recall, and F1 scores have been achieved for a named entity recognition model using bi-directional LSTM-CNNs [3]. Similarly, papers have described success at using LSTMs with character-level features for entity-level sentiment analysis [4]. Services such as Google Cloud also offer APIs in the realm of natural language processing including entity associated sentiment detection that SentiNel aims to achieve [5].

4 Data and Processing

The following publicly available datasets were used:

- **Sentiment140**: contains a collection of 1.6 million short sentences, of which 100,000 were randomly selected for our use in developing our entity analysis model [6].
- **Sentiment Treebank**: contains 11,000+ single sentences extracted from Rotten Tomato reviews for entity detection [2].

To pre-process the datasets for our use, we:

- Manually labelled the entities with their associated sentiments for around 300 sentences.
- Wrote a script that used Google Cloud's Natural Language Processing Sentiment Entity Detection API to label entities for the associated with the sentiment for the remaining, and to verify the manually labelled data [].
- Dropped entries with no detected entities from the Sentiment Treebank dataset.
- For the sentiment analysis model, split 100,000 sentences taken from Sentiment140 using a 90/5/5 ratio into the training, validation, and test set respectively.
- Extracted a small subset of each training set for overfitting.

5 Baseline Model

For the baseline, we used a simple Multi-Layer Perceptron model which averaged the word embedding and then inputted it into one fully connected linear layer. This baseline was better than random guessing (¿50% accuracy), and with some adjustments, it could return much better results. A visual representation follows:

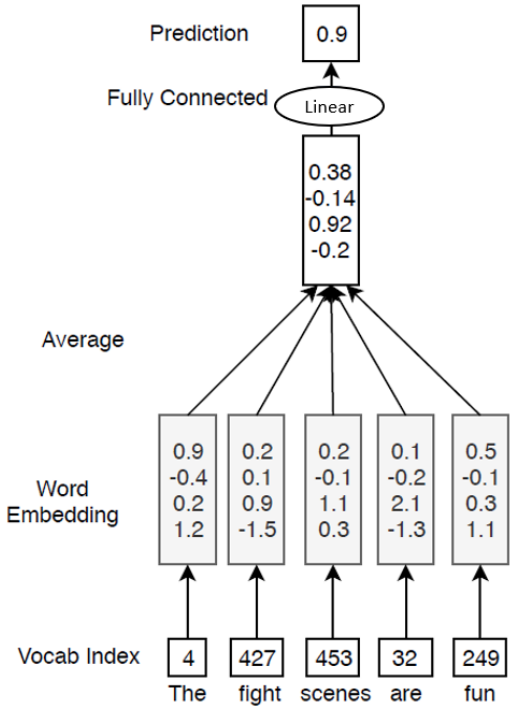

Figure 3: Visual Representation of the baseline model (ECE324 Assignment 5)

6 Attempted Models for Entity Analysis

6.1 Baseline Entity Detection

In order to perform a baseline overfit model, we used 1000 simple sentences with one entity that was usually at the start of each sentence. By predicting the entity to be the first word, the accuracy appeared high, but clearly this was overfit to the point of being heuristic.

6.2 Other Architectures

To facilitate actual learning, following models were coded for entity analysis on our actual training set, but were not able to be tested due to challenges in debugging the training loop that were unresolved at the time of writing:

- CNN
- RNN with GRUs
- LSTM RNN
- Bi-directional LSTM RNN

All of the above were programmed to predict on a 0 to 1 scale whether or not each word in an input sentence was an entity or not. In place of the non-functioning models, the Google Cloud NLP API was used for entity detection.

7 Final Architectures

To improve on the baseline sentiment model, we built a Convolutional Neural Network. This greatly improved our results, which will be discussed later. For experimental purposes, we also coded a Recurrent Neural Network with Gated Recurrent Units which did not work as well as the Convolutional Neural Network on the smaller dataset of 10,000 words. At the time of writing, training it on the larger dataset of 100,000 sentences with Google Colab GPU was causing unresolved errors.

7.1 Sentiment Analysis CNN

Initially, we were training the Sentiment Model with 10,000 sentences (5000 sentences from Sentiment140 and Sentiment Treebank each). To train on this initial dataset, we built a CNN with 2 convolutional layers with 50 kernels each and had kernel sizes of [2,4] respectively, with Batch Normalization at the end of each layer. We then fed it into two fully connected linear layers. The first layer took 100 inputs and output 50, while the second layer took 50 inputs and had 1 output.

Since this model was resulting in both underfitting and overfitting, we changed our dataset for sentiment modeling to contain 100,000 sentences from Sentiment140 only. Experimentally, we also determined that kernel sizes of [2,3] work best for the larger model. Since our validation loss kept increasing in the previous iteration of our CNN with the smaller dataset, we added Dropout to reduce overfitting and as a regularization technique. Dropout randomly zeroed 50% of the neurons in the first layer and 20% of the neurons in the second layer.

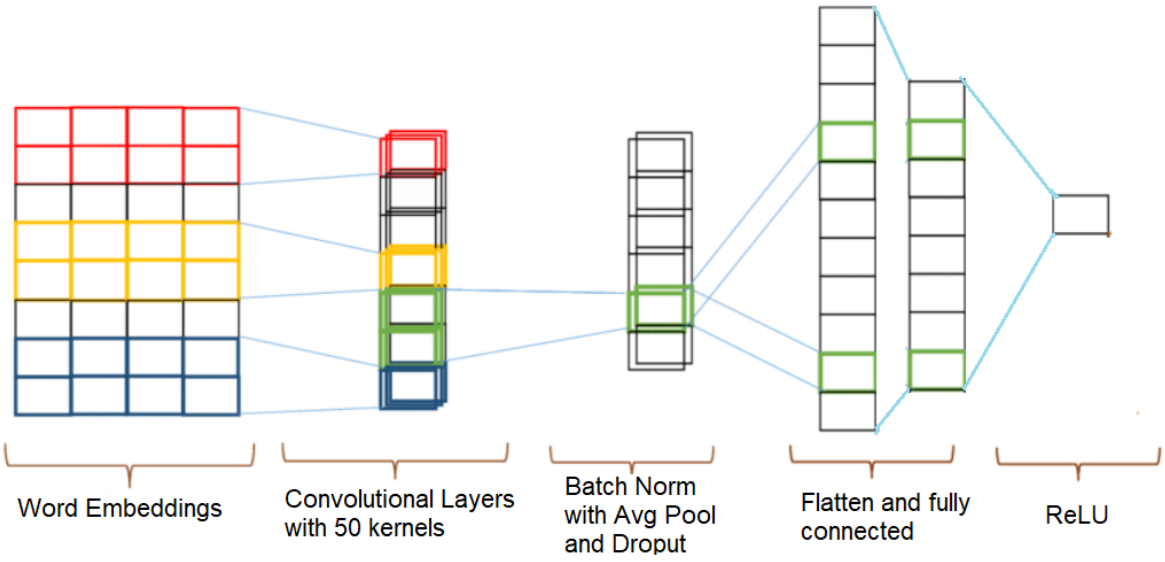

Figure 4: Visual Representation of the final CNN model

7.2 Sentiment Analysis RNN

When we were working with the smaller dataset containing sentences from both Sentiment140 and Sentiment Treebank, we predicted that an RNN might give us a better result because the CNN was both underfitting and overfitting.

So, for this RNN, we used a GRU as the basic cell. The Bucket Iterator was used to divide the dataset into batches as the sentences varied vastly in length. The sentences from Sentiment140 are less than 140 characters while the ones in Sentiment Treebank are much longer and descriptive. Lastly, we used Pack-Padded-Sequence function to pack the word embeddings together in a batch.

8 Quantitative Results

With the small dataset containing 10,000 sentences from Sentiment140 and Sentiment Treebank datasets, the following is the result we obtained from the CNN and RNN. The figure below shows the results obtained from the CNN using the small dataset which is underfitting (training accuracy only reached 60% over 500 epochs).

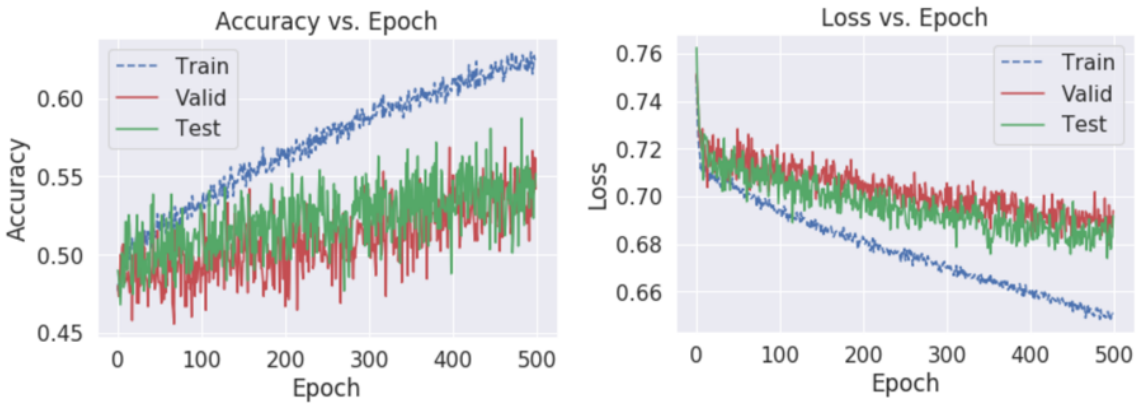

Figure 5: Accuracy and Loss Plots from the CNN with Small Dataset (Train Acc = 0.64, Valid Acc = 0.53)

The figure below shows the results obtained from the RNN using the small dataset which is overfitting as the validation loss is increasing.

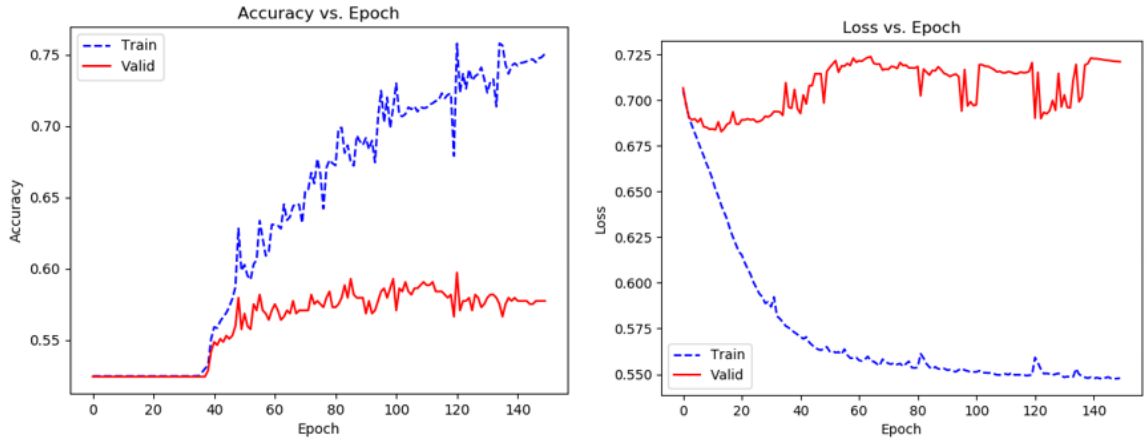

Figure 6: Accuracy and Loss Plots from the RNN with Small Dataset (Train Acc = 0.75, Valid Acc = 0.57)

After switching to our big dataset of 100,000 sentences from Sentiment140 only, the following are the results obtained from our Baseline and CNN.

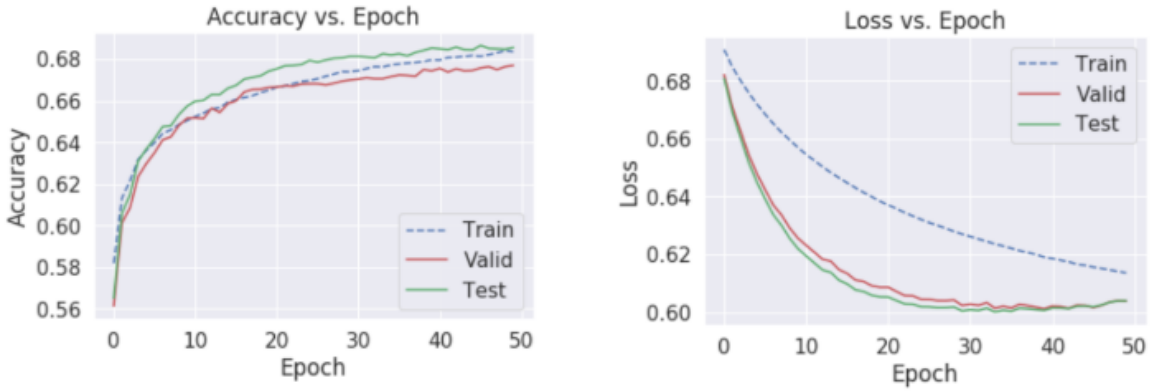

Figure 7: Accuracy and Loss Plots from the Baseline with Large Dataset (Train Acc = 0.68, Valid Acc = 0.67)

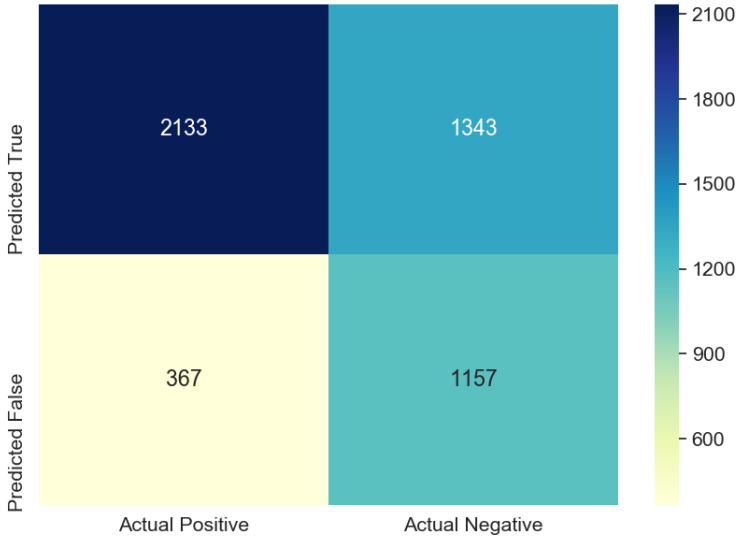

Figure 8: Confusion Matrix for the Baseline Model with Large Dataset

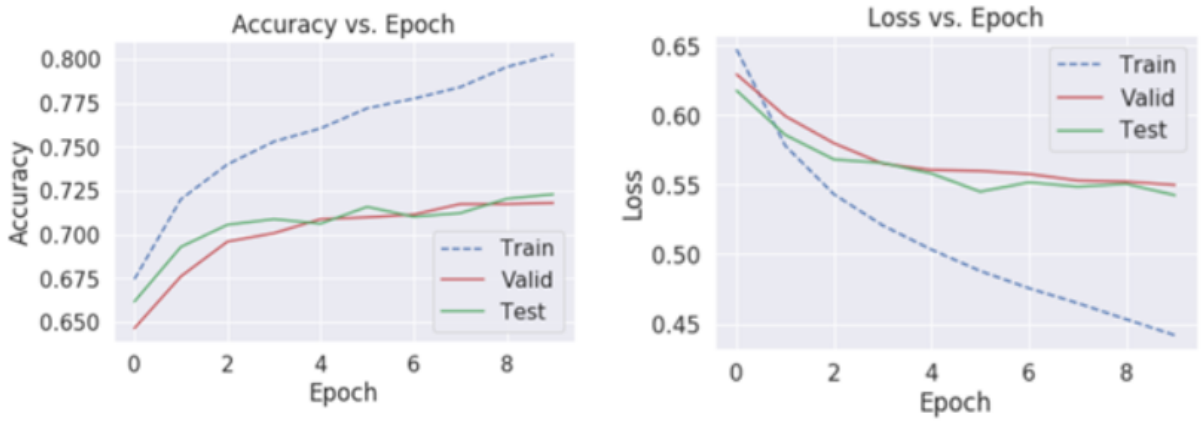

Figure 9: Accuracy and Loss Plots from the CNN with Large Dataset (Train Accuracy = 0.80, Valid Accuracy = 0.72)

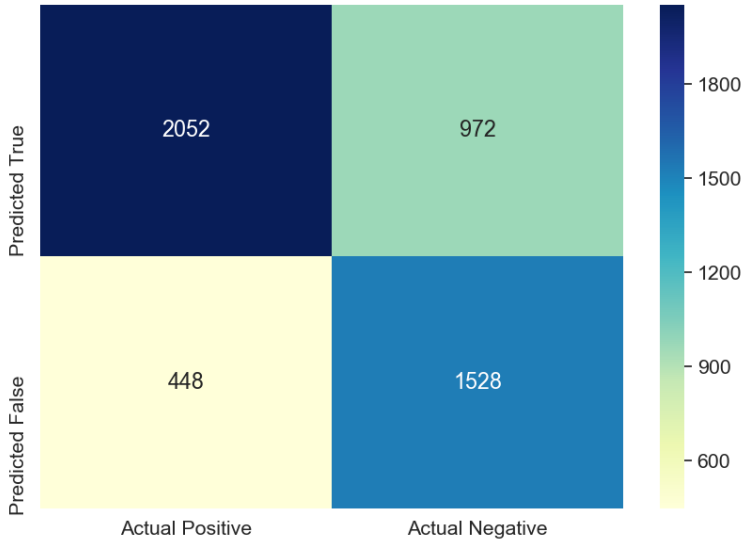

Figure 10: Confusion Matrix for the CNN Model with Large Dataset

Since we were now using 100x more sentences, we switched to Google Colab to train the model. However, Google Colab kept giving us a GPU-related error for the Pack Padded Sequence function which we were not able to solve, even after researching it for two days.

9 Qualitative Results

Using the CNN model trained over 100,000 sentences, we created a bot that detects the polarity and the entity (using Google NLP API), given an input sentence with one entity. The following are some examples of the implementation of this project through the bot. Although the model was trained using negative and positive (0 and 1) labels only, for the bot, we defined the following ranges for the sentiment:

- 0.0 - 0.4: Negative

- 0.4 - 0.6: Neutral
- 0.6 - 1.0: Positive

```
Enter a sentence:
Effective but too tepid biopic
Entity from Google Chrome API: biopic
Sentiment Score: 0.482
Sentiment CNN: neutral
```

Figure 11: Both positive and negative sentiment detected as neutral

```
Enter a sentence:
Illuminating if overly talky documentary
Entity from Google Chrome API: documentary
Sentiment Score: 0.744
Sentiment CNN: positive
```

Figure 12: Both positive and negative sentiment detected as positive

```
Enter a sentence:
I hate Twitter so much
Entity from Google Chrome API: Twitter
Sentiment Score: 0.325
Sentiment CNN: negative
```

Figure 13: Negative sentiment detected correctly

```
Enter a sentence:
It so beautiful outside
Entity from Google Chrome API: outside
Sentiment Score: 0.892
Sentiment CNN: positive
```

Figure 14: Positive sentence detected correctly

10 Discussion

Overall, our final CNN model performs quite well on sentences that are only positive or only negative. With sentences that have both positive and negative sentiments, the model either outputs neutral or positive. Combining these results with high number of False Positive values in the confusion matrices, it is apparent that the model is biased toward being positive. To improve on that, we can augment our dataset to have more negative sentences. Also, we can use a Python library that switches some words in the sentences with its synonym, which will increase the variety of vocabulary present and decrease overfitting.

Something very unusual that happened is that for the Baseline model, our test accuracy is slightly higher than training (Figure 5). One reason for this behaviour could be that while doing the train-valid-test split, the function somehow selected similar sentences for the testing dataset.

11 Learning and Challenges

This project presented challenges beyond our original expectations as follows:

- The usage of bi-directional LSTMs for entity analysis required an understanding of natural language processing beyond the course material and assignments. In order to attempt a model that had reasonable results, our work had to be comparable to models described in published papers in recent years as seen in the background section, which we only realized over the course of the project.
- The variable size of the output prediction based on the size of the input sentence made the model very difficult to debug.
- The availability of Google Cloud's sentiment-entity analysis was unknown to us at the start of the project, so a large amount of time was spent in labelling a entities for a small subset of the dataset before using the service to do so [5].
- We only realized the value and speed of Google Colab later on in the project, so intial training took far longer than we had originally anticipated.

Based on these challenges, if we were to do this project again, we would make immediate use of tools like Google Cloud, Google Colab, and transfer learning to speed up dataset labelling and training. By focusing on scoping the project to a smaller problem at the outset so that we can ensure the successful debugging of the model would also be a good idea - for instance, working with sentences that had a single entity.

12 Ethical Considerations

The ethical considerations associated with our project vary vastly on the usage of it, as will be discussed through the reflexive principlism model [7].

12.1 Stakeholders

Key stakeholders of a functional SentiNel application for Twitter would be the social media users who apply the model to what they read, writers of content/comments on social media, the person entities mentioned in the comments and outlined by the application, and social media companies should they choose to use this model for moderating.

12.2 Autonomy and Justice

If users utilize an application with this model on social media by choice, freedom of expression remains intact. However, if this model were to be used for censorship, this may restrict autonomy but improve justice in the case of circumstances like preventing cyberbullying and hate speech.

12.3 Benevolence and Non-maleficence

This model, when used by choice, can improve the social media experience by allowing users to rethink how they write comments and posts that may be hurtful to others. At the same time, this model when used against freedom of expression forcefully can cause harm to fundamental rights.

13 References

[1] Zhang, M., Zhang, Y., Vo, Duy-Tin "Gated Neural Networks for Targeted Sentiment Analysis," Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).

[2] Baly, R., Hajj, H., Habash, N., Shaban, K. B., El-Hajj, W. (2017). A Sentiment Treebank and Morphologically Enriched Recursive Deep Models for Effective Sentiment Analysis in Arabic. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 16(4), 1–21. doi: 10.1145/3086576

[3] J. P. Chiu and E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[4] Z. Huang, W. Xu, K. Yu, "Named Entity Recognition with Bidirectional LSTM-CNNs," *ArXiv.org*, 2015. doi: arXiv:1508.01991v1 [cs.CL]

[5] "Quickstarts — Cloud Natural Language API — Google Cloud," Google. [Online].
Available: https://cloud.google.com/natural-language/docs/quickstarts. [Accessed: 16-Nov-2019].

[6] For Academics - Sentiment140 - A Twitter Sentiment Analysis Tool. (n.d.). Retrieved from http://help.sentiment140.com/for-students.

[7] J. Beever and A. O. Brightman, "Reflexive Principlism as an Effective Approach for Developing Ethical Reasoning in Engineering," *Science and Engineering Ethics*, vol. 22, no. 1, pp. 275–291, 2015.

Permissions:

- Source Code: Yes
- Project Report: Yes
- Video: We would like to wait till we see it

s