Machine Learning Engineer Nanodegree

Capstone Project

You Are What You Tweet: Detecting Depression in Social Media via Twitter Usage

Anne Bonner
August 3, 2019

I. Definition

Project Overview

More than 300 million people suffer from depression and only a fraction receive adequate treatment. Depression is the leading cause of disability worldwide and nearly 800,000 people every year die due to suicide. Suicide is the second leading cause of death in 15-29-year-olds.[1] Diagnoses (and subsequent treatment) for depression are often delayed, imprecise, and/or missed entirely.

It doesn't have to be this way. Social media provides an unprecedented opportunity to transform early depression intervention services, particularly in young adults.

Every second, approximately 6,000 Tweets are tweeted on Twitter, which corresponds to over 350,000 tweets sent per minute, 500 million tweets per day and around 200 billion tweets per year. [2] Pew Research Center states that currently, 72% of public uses some

---

1 https://www.who.int/news-room/fact-sheets/detail/depression

2 https://www.internetlivestats.com/twitter-statistics/

type of social media. [3] This project captures and analyses linguistic markers associated with the onset and persistence of depressive symptoms in order to build an algorithm that can effectively predict depression. By building an algorithm that can analyze Tweets exhibiting self-assessed depressive features, it will be possible for individuals, parents, caregivers, and medical professionals to analyze social media posts for linguistic clues that signal deteriorating mental health far before traditional approaches currently do. Analyzing linguistic markers in social media posts allows for a low-profile assessment that can compliment traditional services and would allow for a much earlier awareness of depressive signs than traditional approaches.

While many previous studies of depression-related social media posts have relied on Facebook, fewer have focused on Twitter content. However, there are key differences between the two platforms that lead to uniquely different content, and Twitter content is uniquely valuable. While Facebook users frequently post under their real names to friends and family members, Twitter users often use pseudonyms and are more likely to be connected with users they have never met. This allows for a more anonymous means of communication, which may provide a less biased account of an individual's thoughts and experiences. Twitter also provides a broad population of users who may not ordinarily be likely to participate in mental-health related research as well as a broad spectrum of publicly available information, offering a way to overcome some of the limitations of traditional methods of data collection.

I am pursuing the concept that linguistic markers in Tweets may be used to construct statistical models that can detect and even predict depression in ways that can complement and extend traditional approaches to diagnosis.[4] Changes in language and activity have been consistently correlated with changes in activity on social media.[5] Emotional language and linguistic features used in social media posts have been proven to indicate feelings that characterize major depression.[6] People consistently use social media platforms like Twitter to share their thoughts and opinions in written form. Posts on Twitter are particularly useful for analysis because they are most commonly made in the course of daily activities and happenings, which provides a rich and consistent means for capturing behavioral attributes that are relevant to an individual's thinking, mood, socialization, communication, and activities. Because depression is an illness that

[3] https://www.pewinternet.org/fact-sheet/social-media/

[4] https://www.nature.com/articles/s41598-017-12961-9

[5] http://www.munmund.net/pubs/icwsm_13.pdf

[6] https://arxiv.org/pdf/1804.07000.pdf

so often requires the self-reporting of symptoms, social media posts provide a rich source of data and information that can be used to train an efficient model.

Problem Statement

Over the last few years, there has been growing interest in using social media as a tool for public health, ranging from identifying the spread of flu symptoms to building insights about diseases based on Twitter posts.[7] However, research on using social media for analyzing behavioral health disorders is still in its infancy. Park et al.,[8] found evidence that people post about their depression and even their treatment on social media. Eichstaedt, et al., found that Facebook language can predict depression in medical records.[9] De Choudhury et al.,[10] examined linguistic and emotional correlates for postnatal changes of new mothers and built a statistical model to predict extreme postnatal behavioral changes using only prenatal observations. Reece, et al., developed computational models to predict the emergence of Post-Traumatic Stress Disorder in Twitter users.[11] This highlights the potential of social media as a source of signals about likelihood of current or future episodes of depression.

With this project, I am working to expand the scope of social media-based mental health measures and use existing research that has proven the correlation between depression and specific linguistic features in order to build an algorithm that can predict text-based signs of depression. [12] [13] By analyzing linguistic markers, including the use of relevant words associated with depression, it is possible to create a model that can give an individual insight into his or her mental health and well being.

---

[7] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0012948

[8] https://pdfs.semanticscholar.org/8dd5/8913bd343f4ef23b8437b24e152d3270cdaf.pdf

[9] https://www.pnas.org/content/115/44/11203

[10] https://www.microsoft.com/en-us/research/publication/predicting-postpartum-changes-emotion-behavior-via-social-media/

[11] https://www.nature.com/articles/s41598-017-12961-9

[12] https://www.groundai.com/project/utilizing-neural-networks-and-linguistic-metadata-for-early-detection-of-depression-indications-in-text-sequences/

[13] http://www.munmund.net/pubs/icwsm_13.pdf

Datasets and Inputs

In order to build a depression detector, there were two kinds of tweets that were needed for this project: random tweets that do not necessarily indicate depression and tweets that demonstrate that the user may have depression and/or depressive symptoms. A dataset of random tweets can be sourced from the Sentiment140 dataset available on Kaggle [14], however for this binary classification model, the dataset which utilizes the Sentiment140 dataset and offers a set of binary labels proved to be the most effective [15] for building a robust model. There are no publicly available datasets of tweets indicating depression, so "depressive" Tweets were retrieved using the Twitter scraping tool TWINT. The scraped Tweets were manually checked for relevance (for example, Tweets indicating emotional rather than economic or atmospheric depression) and Tweets were cleaned and processed. Tweets were collected by searching for terms specifically related to depression, specifically to lexical terms as identified in the unigram by De Choudhury, et. al. [16]

Because the nature of social media content poses serious challenges to applications of sentiment analysis, VADER was also utilized for general sentiment analysis of Tweets. VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media for general sentiment analysis that is specifically attuned to sentiment in microblog-like contexts. It allows for not only the classification of sentiment, but also the associated sentiment intensity measures.[17] This is extremely useful because Tweets often contain multiple sentiments. VADER doesn't require training data, but is constructed from a human-curated, valence-based, generalizable sentiment lexicon which is fast enough to be used with streaming data. While VADER does not detect depression in text, it gives a foundation for understanding the general sentiment of the data.

Metrics

The accuracy of the model was evaluated and compared to a binary classification baseline model using logistic regression. The models were analyzed for accuracy and a

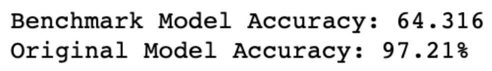
[14] https://www.kaggle.com/kazanova/sentiment140

[15] https://www.kaggle.com/ywang311/twitter-sentiment/data

[16] https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/viewFile/6124/6351

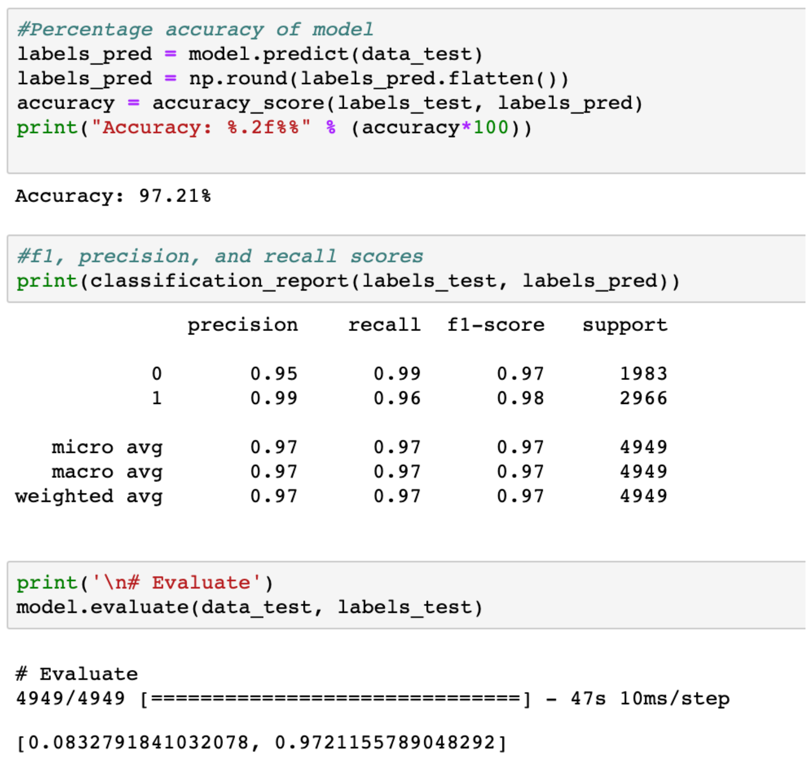
[17] http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf

classification report was run to determine precision and recall scores. The data were split into training, testing, and validation sets and the accuracy for the model was determined based on the model's performance with the testing data, which were kept separate. While the performance of the benchmark logistic regression model was 64.32% using the same data, learning rate, and epochs, the LSTM model performed significantly better at 97.21%.



```
Benchmark Model Accuracy: 64.316
Original Model Accuracy: 97.21%
```

Precision, recall, accuracy and f1 scores were run to evaluate the effectiveness of the original model.



```
#Percentage accuracy of model
labels_pred = model.predict(data_test)
labels_pred = np.round(labels_pred.flatten())
accuracy = accuracy_score(labels_test, labels_pred)
print("Accuracy: %.2f%%" % (accuracy*100))
```

```
Accuracy: 97.21%
```

```
#f1, precision, and recall scores
print(classification_report(labels_test, labels_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1983
           1       0.99      0.96      0.98      2966

   micro avg       0.97      0.97      0.97      4949
   macro avg       0.97      0.97      0.97      4949
weighted avg       0.97      0.97      0.97      4949
```

```
print('\n# Evaluate')
model.evaluate(data_test, labels_test)
```

```
# Evaluate
4949/4949 [==============================] - 47s 10ms/step

[0.0832791841032078, 0.9721155789048292]
```

II. Analysis

Data Exploration

In order to build a depression detector, I used two kinds of tweets that: random tweets that do not necessarily indicate depression and tweets that demonstrate that the user may have depression and/or depressive symptoms. The random tweets dataset is available on Kaggle and classifies 1.6 million texts into binary categories.[18] There are no publicly available datasets of tweets indicating depression, so "depressive" Tweets, or Tweets that contain linguistic markers indicative of depression, were retrieved using the Twitter scraping tool TWINT.

Tweets were searched and scraped according to terms as specified in research by De Choudhury, et al.[19] The Tweets were gathered in a random 24-hour period and saved as separate csv files. Information that could identify the Twitter user was removed, including username, name, conversation_id, created_at, and place, and geolocation. Additionally, timezone, likes count, links, retweets, quote urls, videos, photos, user retweet id, replies count, and retweet counts were dropped as they were unnecessary for the classifier. Null values were removed as well. Data was gathered for the terms:

- Depressed
- Depression
- Hopeless
- Lonely
- Suicide
- Antidepressant
- Antidepressants

These Tweets proved to contain lexical features strongly indicative of depression and were ideal for training an efficient and robust classifier.

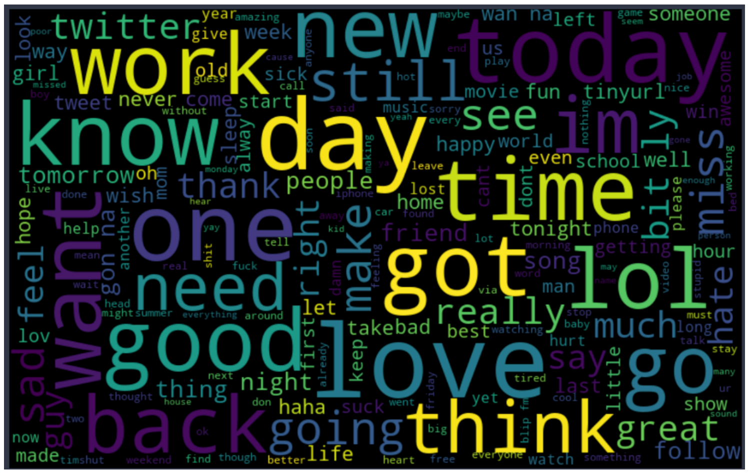
[18] https://www.kaggle.com/ywang311/twitter-sentiment/data

[19] https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/viewFile/6124/6351

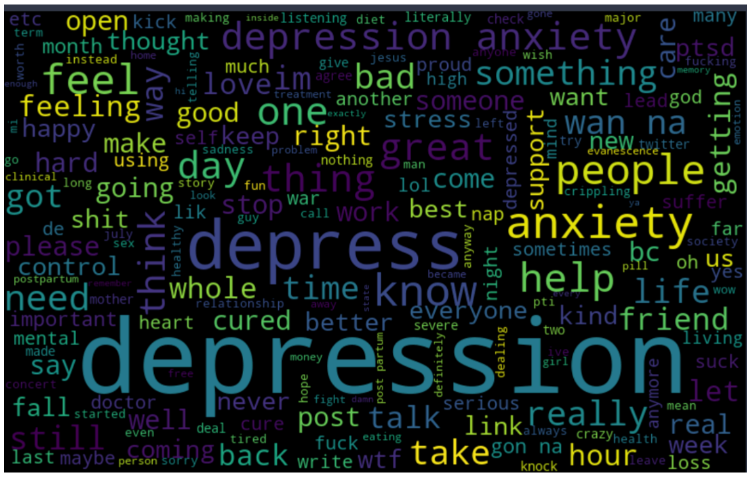
Exploratory Visualization

Once the Tweets were cleaned, it was easy to see the difference between the two datasets by creating a word cloud with the cleaned Tweets. With only an abbreviated TWINT Twitter scraping, the differences between the two datasets were clear:

Random Tweet Word Cloud:



Depressive Tweet Word Cloud:



The word clouds initially contained days of the week and the names of months, so it was necessary to filter out those results in the data cleaning and preprocessing phase.

Algorithms and Techniques

The scraped Tweets were manually checked for relevance and Tweets were cleaned and processed by expanding contractions, removing links, images, hashtags, mentions, emojis, stop words, and punctuation. In order to get a sense of the general sentiment of the Tweets, VADER was used for exploratory sentiment analysis. VADER analysis offers nuanced scores and includes a compound score, which is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(extreme negative) and +1(extreme positive).

Typical threshold values, as cited on the VADER Sentiment Wiki [20] are:

1. Positive sentiment: compound score >= 0.05

2. Neutral sentiment: (compound score > -0.05) and (compound score <0.05)

3. Negative sentiment: compound score <=-0.05

Because this project uses binary classification, scraped Tweets were given binary classification labels. After further research and analysis, it became clear that labeling the Tweets as "positive" with any VADER analysis of sentiment analysis > 0 and "negative" with any sentiment analysis result <= 0 was not the most effective way to label the data for an algorithm designed to detect depression. In fact, Tweets that appeared to be "neutral" exhibited as many, if not more, linguistic features of depression than Tweets labeled as "negative," so Tweets were relabelled with sentiment analysis results > 0.05 as "positive" and <= 0.05 as "negative."

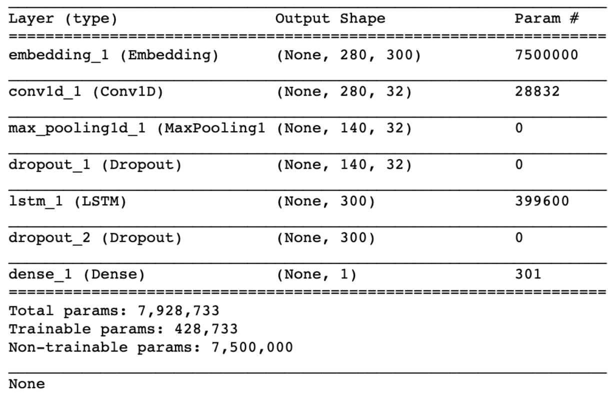
Words were stemmed using a porter stemmer. A tokenizer was utilized to assign indexes to words and filter out infrequent words. An embedding matrix was created for the embedding layer. The data was split into 60% training, 20% testing, and 20% validation sets and randomly shuffled.

The LSTM model takes in the tokenized tweets and feeds them into an embedding layer to get an embedding vector. The model takes in an input and then outputs a number representing the probability that the Tweet indicates depression. The model takes in each input Tweet, replaces it with its embeddings, and then runs the new embedding

[20] https://github.com/cjhutto/vaderSentiment?source=post_page---------------------------#about-the-scoring

vector through a convolutional layer, which is well-suited for learning spatial structure from data. The output of the LSTM layer is fed into a dense model for prediction.

Besides the LSTM layer, the model has an embedding layer, a convolutional layer, and a Dense layer and uses max pooling, a dropout of 0.5 in the LSTM layer and a dropout of 0.5 in the convolutional layer, binary cross entropy loss, Nadam optimizer, and a ReLu activation function in the first layer and a sigmoid activation function in the dense layer. Accuracy, precision, f1, and loss are calculated, recorded, visualized and compared to a a benchmark logistic regression model.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 280, 300)          7500000   
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 280, 32)           28832     
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 140, 32)           0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 140, 32)           0         
_________________________________________________________________
lstm_1 (LSTM)                (None, 300)               399600    
_________________________________________________________________
dropout_2 (Dropout)          (None, 300)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 301       
=================================================================
Total params: 7,928,733
Trainable params: 428,733
Non-trainable params: 7,500,000
_________________________________________________________________
None
```

Benchmark

Because of the nature of mental illness and its subjectivity, it made sense to use a binary classification model. The benchmark model chosen was a logistic regression model. The benchmark model was trained with the same training data and number of epochs as the original model. The accuracy of the original model was evaluated and compared to the benchmark model. The models were analyzed for accuracy and a classification report was run to determine precision and recall scores.

III. Methodology

Data Preprocessing

For the model, both random Tweets and Tweets indicating depression were necessary. Tweets indicating depression were retrieved using the Twitter scraping tool TWINT.

The scraped Tweets and the precise steps taken to prepare the dataset of scraped Tweets can be found in the “TWINT_cleaning_final” notebook in the “TWINT_data” folder. Tweets were searched and scraped according to terms indicative of depression as specified in research by De Choudhury, et al. [21] The Tweets were gathered in a random 24-hour period and saved as separate csv files according to search terms and then combined, resulting in a dataset with 224,273 Tweets. Stop words were removed with NLTK and null values were removed as well. Once the null values were removed, it was possible to create an exploratory VADER sentiment analysis score to run an overall sentiment analysis of the tweets, which was calculated and stored. Information that could identify the Twitter user was dropped, including username, name, conversation_id, created_at, and place, and geolocation. Additionally, timezone, likes count, links, retweets, quote urls, videos, photos, user retweet id, replies count, and retweet counts were dropped as they were unnecessary for the classifier.

Before the data could be used in the model, it was necessary to expand contractions, remove links, hashtags, capitalization, and punctuation. Negations needed to be dealt with, which entailed creating a dictionary of negations so that negated words could be effectively handled. Links and urls needed to be removed along with whitespaces. Additionally, stop words beyond the standard NLTK stop words needed to be removed to make the model more robust. These words included days of the week and their abbreviations, month names, and the word “Twitter,” which surprisingly showed up as a prominently featured word when the word clouds were created. The tweets were then tokenized and PorterStemmer was utilized to stem the tweets.

[21] https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/viewFile/6124/6351

Implementation

Mental health, specifically depression, is a subjective and complex topic. While it may be possible to quantify the degree to which one might be depressed based on a Tweet, the only real question necessary for this making this classifier as robust as possible is, **is an individual exhibiting linguistic markers indicative of depression?** With subjective data and a question that demands a binary answer, a binary classification model made the most sense for this project.

While a logistic regression model was an acceptable choice for a benchmark model, a Long Short Term Memory network (LSTM) model was idea for the project at hand. A LSTM is capable of learning long-term dependancies and works incredibly well on a large variety of problems. While RNNs in general can have difficulty with long-term dependencies, LSTMs are explicitly designed to avoid the long-term dependency problem. Adding a convolutional layer improved the model substantially. Convolutional neural networks (CNNs) are well-suited for learning spatial structure from data and learns structure from the sequential data which it passes into a LSTM layer.

Utilizing Word2vec [22] [23] [24] also made the model more robust. Word2vec turns text into a numerical form that deep nets can understand. It groups the vectors of similar words together in vectorspace and is able to detect similarities mathematically. The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net.

The LSTM model takes in the tokenized tweets and feeds them into an embedding layer to get an embedding vector. The model takes in an input and then outputs a number representing the probability that the Tweet indicates depression. The model takes in each input Tweet, replaces it with its embeddings, and then runs the new embedding vector through a convolutional layer, which is well-suited for learning spatial structure from data. The convolutional layer takes advantage of this and learns structure from the sequential data, which it then passes into a standard LSTM layer. The output of the LSTM layer is fed into a dense model for prediction. The model has an embedding layer, a convolutional layer, and a dense layer and uses max pooling, a dropout of 0.5 ***********, binary cross entropy loss, a Nadam optimizer, and also a ReLU activation

[22] https://arxiv.org/pdf/1301.3781.pdf

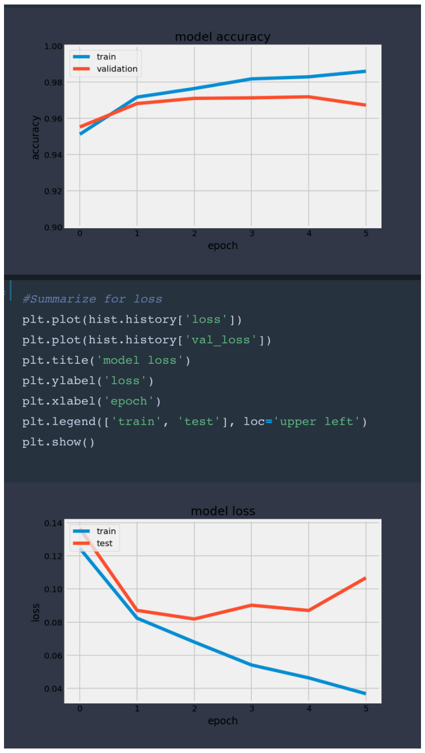
[23] https://radimrehurek.com/gensim/models/word2vec.html

[24] https://skymind.ai/wiki/word2vec

function in the first layer, and a sigmoid activation function in the dense layer. Accuracy and loss are recorded and visualized and compared to a a benchmark logistic regression model.

Refinement

Early in the process, it became clear that the most important part of refining the model to get more accurate results would be the data gathering, cleaning, and preprocessing stage. Until the Tweets were appropriately scraped and cleaned, the model had unimpressive accuracy. By cleaning and processing the Tweets with more care, the robustness of the model improved substantially. After that, it became necessary to adjust the optimizer and dropout. While I originally had a dropout of 0.2 in both the convolutional and LSTM layers, increasing the dropout to 0.5 in the convolutional layer and 0.5 in the dropout layer improved the model by more than 3% and reduced the appearance of overfitting, as did adjusting the parameters of the optimizer.



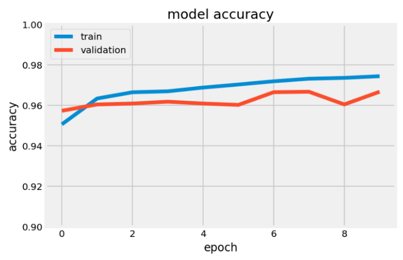
```
#Summarize for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

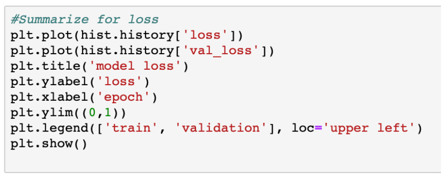
Graphs above display results indicative of overfitting.

Adjusting not only the dropout, but the batch size as well led to improved model performance. Once the complexity of the model was reduced, dropout was increased, batch size was reduced, and the model was rerun in order to examine the results.

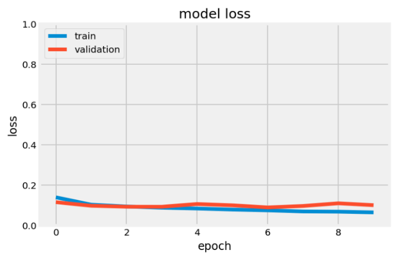
Graphs representing stages of model improvement

1.

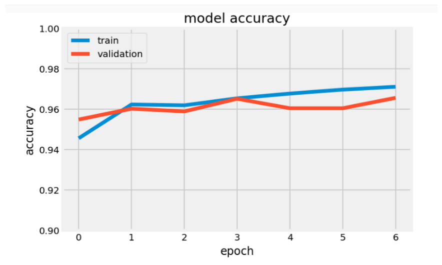


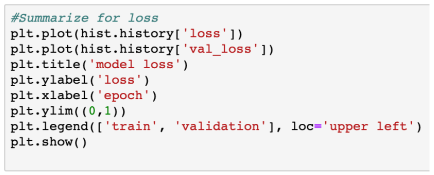


```
#Summarize for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.ylim((0,1))
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

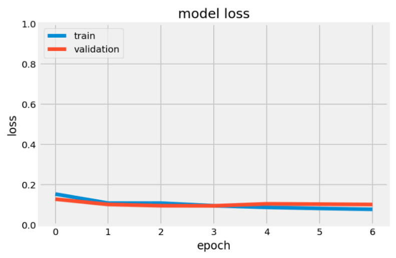


2.

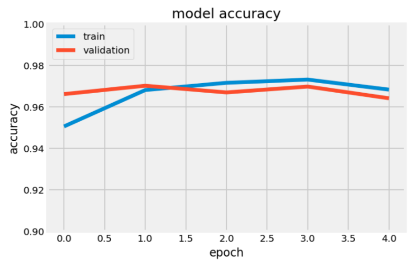


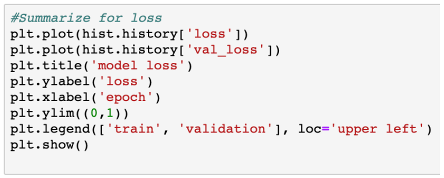


```
#Summarize for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.ylim((0,1))
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

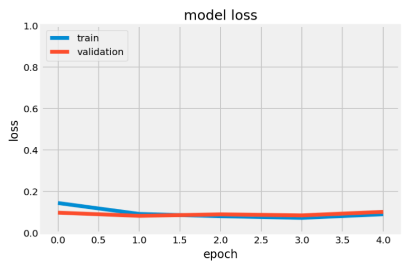


3.

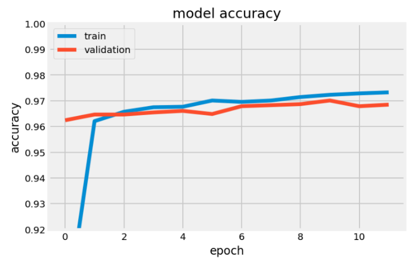


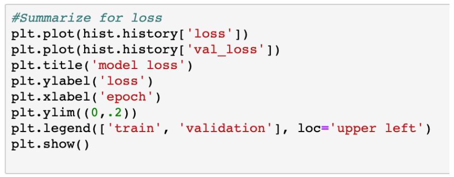


```
#Summarize for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.ylim((0,1))
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

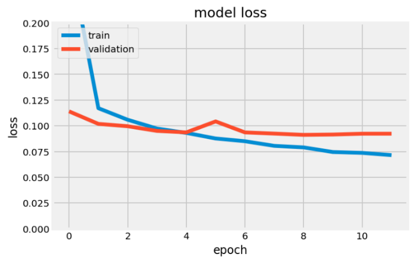


4.

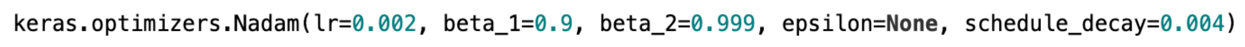




```
#Summarize for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.ylim((0,.2))
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

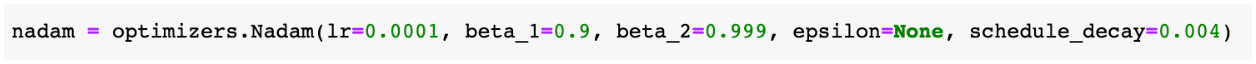


Default Nadam parameters [25]*:*



```
keras.optimizers.Nadam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, schedule_decay=0.004)
```

Adjusted Nadam parameters:



```
nadam = optimizers.Nadam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None, schedule_decay=0.004)
```

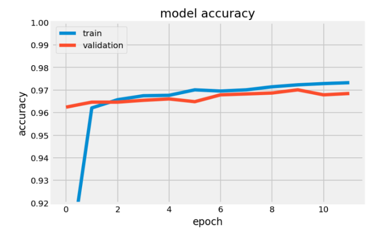
Adding embedding to the model also made it more robust. Keras provides a convenient way to convert positive integer representations of words into a word embedding by an embedding layer. The layer takes arguments that define the mapping, including the maximum number of expected words (also called the vocabulary size, e.g. the largest integer value that will be seen as an integer). The layer also allows you to specify the dimensionality for each word vector (output dimension). Embedding is a technique where words are encoded as real-valued vectors in a high-dimensional space, where the similarity between words in terms of meaning translates to closeness in the vector space. Discrete words are mapped to vectors of continuous numbers. This is useful when working with natural language problems with neural networks and deep learning models which require numbers as input.

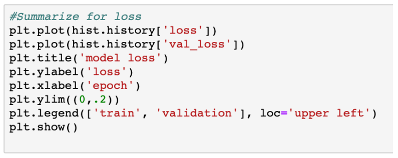
[25] https://keras.io/optimizers/

IV. Results

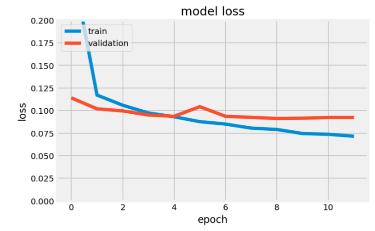
Model Evaluation and Validation

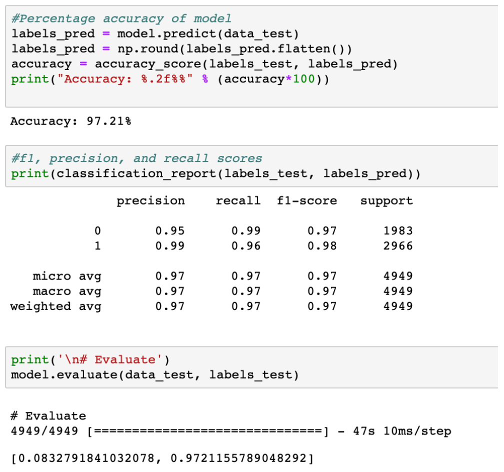
The final architecture, parameters, and hyperparameters were chosen because they performed the best among all the tried combinations. The LSTM model takes in the tokenized tweets and feeds them into an embedding layer to get an embedding vector. The model takes in an input and then outputs a number representing the probability that the Tweet indicates depression. The model takes in each input Tweet, replaces it with its embeddings, and then runs the new embedding vector through a convolutional layer, which is well-suited for learning spatial structure from data. The convolutional layer takes advantage of this and learns structure from the sequential data, which it then passes into a standard LSTM layer. The output of the LSTM layer is fed into a dense model for prediction. The model has an embedding layer, a convolutional layer, and a Dense layer and uses max pooling, a dropout of 0.5, binary cross entropy loss, an Nadam optimizer, and a ReLu activation function in the first layer and a sigmoid activation function in the dense layer. Accuracy and loss are recorded and visualized and compared to a benchmark logistic regression model.





```
#Summarize for loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.ylim((0,.2))
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```





```
#Percentage accuracy of model
labels_pred = model.predict(data_test)
labels_pred = np.round(labels_pred.flatten())
accuracy = accuracy_score(labels_test, labels_pred)
print("Accuracy: %.2f%%" % (accuracy*100))
```

```
Accuracy: 97.21%
```

```
#f1, precision, and recall scores
print(classification_report(labels_test, labels_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1983
           1       0.99      0.96      0.98      2966

   micro avg       0.97      0.97      0.97      4949
   macro avg       0.97      0.97      0.97      4949
weighted avg       0.97      0.97      0.97      4949
```

```
print('\n# Evaluate')
model.evaluate(data_test, labels_test)
```

```
# Evaluate
4949/4949 [==============================] - 47s 10ms/step

[0.0832791841032078, 0.9721155789048292]
```

```
Benchmark Model Accuracy: 64.316
Original Model Accuracy: 97.21%
```

The model was fit on the training and validation sets and testing was performed on the test set, which was kept separate for accuracy of evaluation. The results were summarized for accuracy and loss and plotted and the percentage accuracy of the model was calculated on the testing data. A classification report was also run for f1, precision, and recall scores.

Justification

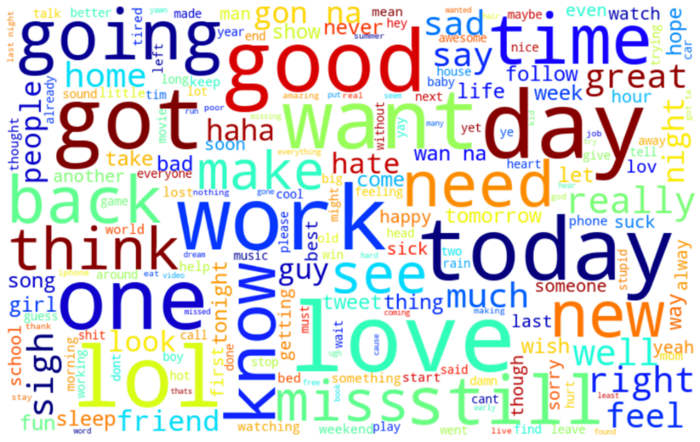
The final model proves to be far more accurate than the benchmark model. The benchmark model, run on the same data for the same number of epochs, shows an accuracy of approximately 64%, while the final model has an accuracy of approximately 97%. This proves to be a much more robust and effective model for depression prediction and it appear that this solution is significant enough to have solved the difficulty of effectively analyzing Tweets for depression.

V. Conclusion

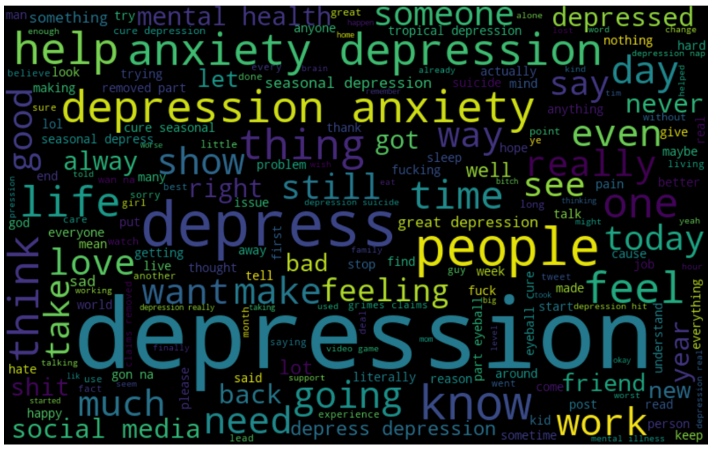
Free-Form Visualization

The use of linguistic markers as a tool in the analysis and diagnoses of depression has enormous potential. Depression can so quickly be seen in text, even without the use of complex models. Simply by collecting, cleaning, and processing available data, visual analysis alone can illuminate the difference between random Tweets and Tweets that have depressive characteristics.

Random Tweets



Tweets with Depressive Characteristics



The potential of linguistic analysis in the arena of mental health cannot be overstated. By analyzing a person's words, you have a clear and valuable window into his or her mental state. Even the simplest analysis of social media can provide us with unprecedented access into individuals thoughts and feelings and lead to substantially greater understanding and treatment of mental health.

Reflection

After deciding to look at the connection between mental health and social media, it became apparent that the most effective way to begin this process would be to focus specifically on one area of mental health and one social media platform. Because depression is such a wide-spread issue, it made the most sense to focus on depression and its associated research. While Facebook has been studied repeatedly and a wealth of information exists on using Facebook posts to detect and analyze mental health, Twitter seemed to me to be a more intriguing platform to use. Tweets are fundamentally different from Facebook posts. They are more anonymous, shorter, faster, and more likely to be made quickly throughout the day and night. In the future, I would like to build this algorithm into an app that can analyze texts for signs of depression, and Tweets are closer in structure to texts sent throughout the day than posts on other platforms. Additionally, Twitter data is readily available as existing datasets and it's simple to scrape and even stream, if desired.

Once deciding that Twitter would be the way to go, finding (or creating) the optimal data was the next issue to tackle. There are existing datasets containing random Tweets and Tweets analyzed for sentiment, but sentiment analysis alone is not the same thing as depression analysis. Because there are no publicly available datasets containing "depressive" Tweets, it became apparent that I would have to create my own. One of the first ways that came to mind was using Tweepy[26], but Tweepy requires a developer account, meaning that a user would need to apply for a Twitter developer account, and I wanted to make sure that this model would be easily reproducible to anyone interested in recreating the model. After further research, TWINT turned out to be the optimal solution. TWINT allows any user the ability to search and scrape Tweets in a variety of ways and collect and store them.

The question at that point became, what would make a Tweet indicative of depression? A number of research papers exist which look at lexical markers of depression, and they

[26] https://www.tweepy.org/

all agree on a number of points. People suffering from depression often subconsciously use very specific linguistic markers that indicate depression. It would be possible to build a dictionary including these terms and give them separate weights, but after reading a number of papers related to depression, mental health, and linguistics, it became clear that depression is something that is most frequently self-reported and, at least initially, self-assessed. Therefore, if a person says that she is depressed, it is extremely likely that she is depressed. Additionally, words like "hopeless," "lonely," and "suicide" are consistently associated with people who are suffering from feelings associated with depression, as are people who reference specific medications, even as generally as "antidepressant." While it would be possible to build dictionaries that include all of the lexical features specific to depression, including specific terms, an increased use of personal pronouns, increased religious references, time-of-day analysis, and so on, it seemed that by scraping Tweets searched by very specific terms, those Tweets might contain all of the lexical features that a good model would need to learn and become both robust and accurate.

After scraping Tweets using TWINT for the terms:

- Depressed
- Depression
- Hopeless
- Lonely
- Suicide
- Antidepressant
- Antidepressants

for a random 24 hour period, the data were combined into a single dataset. The Tweets were manually cleaned (for example, to remove references to economic rather than emotional depression), and then cleaned processed according to the steps listed in the "data cleaning and preprocessing" section above.

The next issue involved the choice of classifier. There are a number of ways to analyze the information, but the reality is that mental health, specifically depression, is a subjective and complex topic. While it may be possible to quantify the degree to which one might be depressed based on a Tweet, the only real question that matters for this project is, is an individual exhibiting linguistic markers indicative of depression? Knowing the question and the subjective nature of mental illness, it was obvious that a binary classification model would make the most sense for this project.

While a logistic regression model made sense as a benchmark model, a Long Short Term Memory network (LSTM) model wound up being the most robust for the project at hand. A recurrent neural network allows information to be passed from one step of a network to another, and are ideal for sequences, lists, and other language processing problems. A LSTM is capable of learning long-term dependancies and work incredibly well on a large variety of problems. While RNNs have difficulty with long-term dependencies, LSTMs are explicitly designed to avoid the long-term dependency problem.

It also made sense to add a convolutional layer. Convolutional neural networks (CNNs) are well suited for learning spatial structure from data and learns structure from the sequential data which it passes into the LSTM layer.

Utilizing Word2vec [27] [28] [29] also made the model much more robust. Word2vec is a two-layer neural net that processes text. Its input is a corpus of text and its output is a set of vectors. It turns text into a numerical form that deep nets can understand. Word2vec groups the vectors of similar words together in vectorspace. It's able to detect similarities mathematically. Given enough data, usage, and contexts, it can make highly accurate guesses about a word's meaning based on past appearances. These guesses can be used to establish a word's association with other words.The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net.

The LSTM + CNN model takes in an input and then outputs a single number representing the probability that the tweet indicates depression. The model takes in each input sentence, replaces it with its embeddings, and then runs the new embedding vector through a convolutional layer. The convolutional layer passes the structure that it learns from the sequential data into a LSTM layer. The output of the LSTM layer is then fed into a Dense model for prediction.

Once the model was designed and built, the issue then became refining the model to achieve the best results. Initially, the model suffered from overfitting, as evidenced in the graph. However, by decreasing the complexity of the model and increasing the dropout, a much more robust model was achieved and evidence of overfitting was eliminated.

---

[27] https://arxiv.org/pdf/1301.3781.pdf

[28] https://radimrehurek.com/gensim/models/word2vec.html

[29] https://skymind.ai/wiki/word2vec

The data for this model were split into 60% training, 20% validation, and 20% testing sets. The testing set was kept separate until the end, ensuring the accuracy of the model. The final results were analyzed by looking at both accuracy and a classification report. The classification report is here:

With accuracy of 97%, this model proves to be an accurate and robust model that should prove effective for further development and use.

Improvement

It would be exciting to take this project further and look not only at linguistic markers for analysis that can generalize to other social media platforms and text messages, but visual cues as well. Work has been done looking at the connection between images posted on social media and depression [30] [31] [32], and it would be exciting to find a way to incorporate that research into this model. People are visual by nature, and a model that could analyze text and images simultaneously could provide an even greater understanding of a person's mental state long before traditional means can.

A model that could accurately analyze both text and images might prove robust enough to take on platforms like Instagram and SnapChat, allowing for the possibility of reaching an even younger demographic, perhaps offering help to youth suffering from the early stages of depression and other mental health issues long before traditional methods of awareness and intervention can.

[30] https://www.upi.com/Health_News/2017/08/08/Instagram-photos-may-help-predict-depression/8801502216966/

[31] https://epjdatascience.springeropen.com/articles/10.1140/epjds/s13688-017-0110-z

[32] https://arxiv.org/abs/1904.02670