Introduction:

Reddit is one of the most visited websites in the world, with more than 48 million active users and pulling in nearly 1.5 billion visits in the month of September in 2020 alone. As such, it is an extremely valuable tool for social media marketing, distributing a viral campaign, or spreading content quickly. Despite this, it is difficult to discern true user sentiment because of the prolific sarcastic culture that is present in this online forum.

Users on Reddit oftentimes self-annotate when they are sarcastic using the tag “/s” at the end of their sarcastic comments. Thanks to the innate behaviors of online communication, Reddit serves as a wealth of internet slang and interactions to understand sarcasm and textual communication.

Using machine learning, I built a model that can correctly diagnose whether or not a comment is sarcastic or not up to a 71% accuracy using deep learning and NLP techniques.

Data cleaning and Preprocessing:

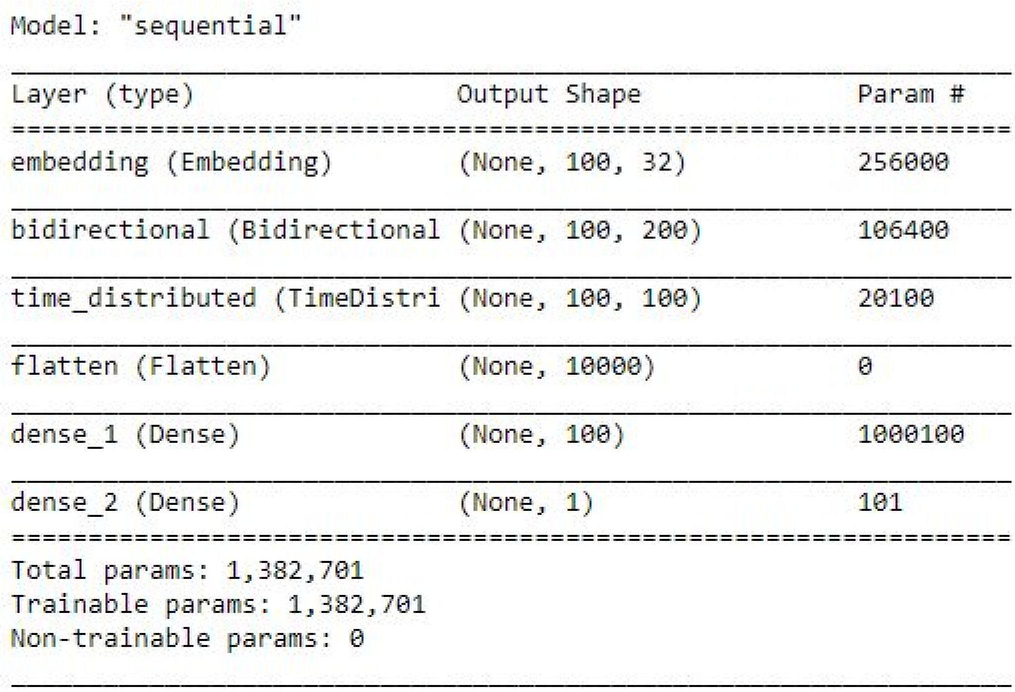
There were 53 null values in the comments and were dropped from the dataset. Because this was negligent in a dataset with over 1 million comments, no accommodations were made in light of this change.

The data was then cleaned by removing punctuations, lowering all capitalizations, and removing stopwords from the NLTK English library.NLP techniques were used to preprocess the data. To remove the morphological affixes from words, the SnowballStemmer (or also known as the Porter2 Stemmer) from the NLTK library was selected as the stemming algorithm as is by convention for most practical uses. For processing the naive bayes and logistic regression models, the comments were transformed by a Tf-Idf vectorizer which at this point separated between a unigram and bigram bag-of-words models. The native keras tokenizer was used for vectorizing with a vocabulary size of 8000, and later padded with a max length of 100 characters to fit into the input layer uniformly. While GloVe embeddings were considered to be used to further increase the deep learning model’s generalizability and potential increase in accuracy, the limitations on computational power constrained that possibility.

Methods:

The deep learning architecture was relatively simple. The word-level LSTM had an embedding layer, a single LSTM layer, and one dense layer.

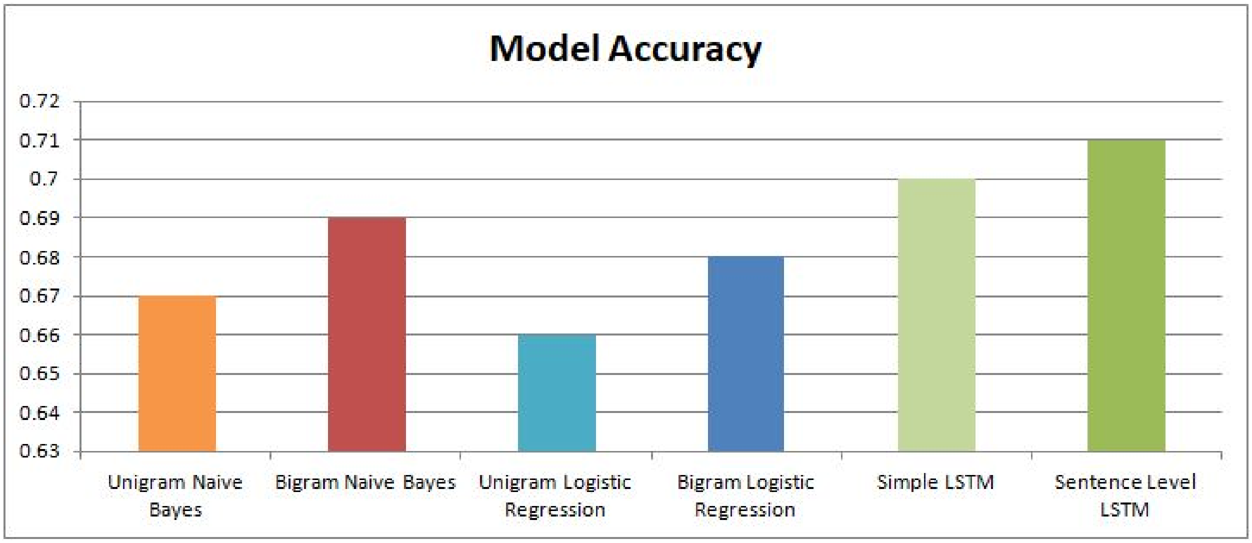
In order for the model to get a sentence-level understanding of each comment, a bidirectional LSTM layer was used, along with a time distributed layer. First the bidirectional layer will save both past and future information which allows for better contextual understanding. In addition, a TimeDistributed layer adds a time dimension as a wrapped layer to each slice of the input tensor. This outputs a processed time series of data tagged with sequential information so that each “past” and “future” is taken in an order of the sentence.



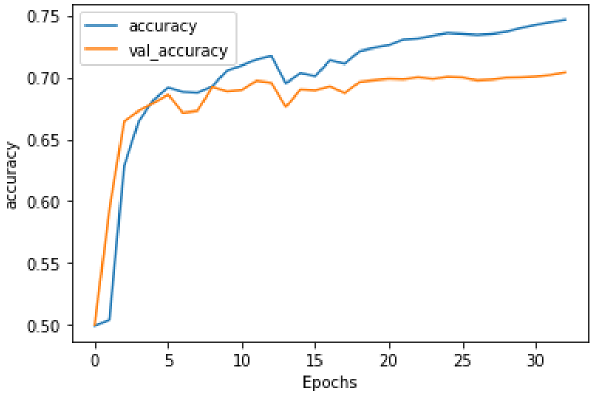
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding (Embedding)        (None, 100, 32)           256000    
_________________________________________________________________
bidirectional (Bidirectional (None, 100, 200)          106400    
_________________________________________________________________
time_distributed (TimeDistri (None, 100, 100)          20100     
_________________________________________________________________
flatten (Flatten)            (None, 10000)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 100)               1000100   
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101       
=================================================================
Total params: 1,382,701
Trainable params: 1,382,701
Non-trainable params: 0
_________________________________________________________________
```

Model Architecture Summary Table

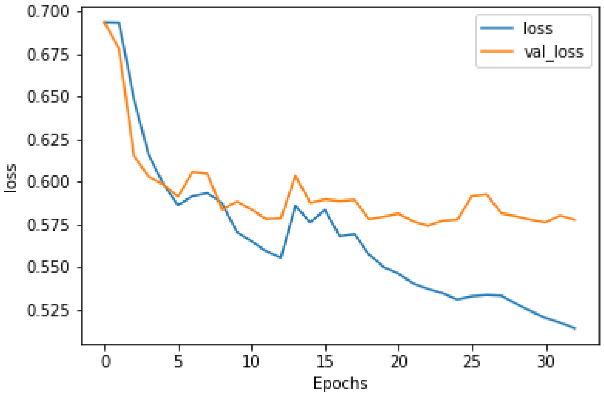
Results:



LSTM Fit Accuracy History



LSTM Fit Loss History



Model Evaluation Table

	Recall	Precision	F Score
Unigram Naive Bayes	0.67	0.67	0.67
Bigram Naive Bayes	0.69	0.69	0.69
Unigram Logistic Regression	0.66	0.66	0.66
Bigram Logistic Regression	0.68	0.68	0.68
LSTM - Simple Net	0.70	0.70	0.70
LSTM - Sentence Level	0.71	0.71	0.71

Discussion

Every model is able to predict significantly higher than the baseline balanced set of 50%. Surprisingly, the macro average of every model evened out the recall and precision W can see that the Bigram Naive Bayes model still performs quite decently in comparison to our neural net model. This simple neural net model is still able to classify sarcasm slightly better, and if I were to further extract information from the dataset along with optimizations and a larger, more complex model I can easily foresee that the deep learning model would be able to reach a significantly higher level of accuracy than the simpler models.

Future Work

1. Capture more information with sentence-context level LSTM to understand context with parent comments or subreddit type.
2. Explore different architectures that concatenate multiple types of data such as the lexical density, score, etc.

Data acquisition:

The data was gathered using the Pushshift API, which is a copy of reddit objects. The data is copied into Pushshift at the time it is posted to reddit. This data was posted on kaggle with a balanced and unbalanced set. The true ratio is about 1:100 with 1.3 million sarcastic statements, however this distributed corpus has about 1 million total comments within the balanced set.

References:

https://www.alexa.com/siteinfo/reddit.com

https://www.statista.com/statistics/443332/reddit-monthly-visitors/#:~:text=Total%20global%20visitor%20traffic%20to%20Reddit.com%202020&text=Reddit%20is%20a%20web%20traffic,the%20most%2Dvisited%20websites%20online.

https://www.statista.com/statistics/248074/most-popular-us-social-networking-apps-ranked-by-audience/