The Life-Cycle of Hate Speech

Sophia Mlawer, Yangzhou Ou, Natalie Ayers

Github Link

Abstract

We develop a research agenda to examine the progression and escalation of hate speech as displayed through digital media. We first identify racist and minority group-based hate speech in Google searches before focusing on the classification of Twitter posts as hate speech. We develop and evaluate numerous classification models, including traditional supervised learning models using a variety of feature types, a bi-directional LSTM model, a convolutional neural network, and a model using distilBERT. We found the highest F1 performance using the bi-LSTM, with an F1 score of 0.9967. We employed this bi-LSTM to predict hate speech in a secondary Twitter corpus. Our evaluation of these predictions and a comparison with the Google Trends temporal patterns indicate a need for a much larger corpus to predict on in order to be more successful identifying linked forms of hate speech.

Introduction

In this paper, we hope to examine the progression and varying expression of hate speech through three lenses: search platforms, social media posts, and occurring events. We hypothesize that the spread of many toxic movements can be represented by the appearance and intensity of the foundational ideas in these outlets. We plan to identify the occurrences of hate speech, particularly related to racial identity, in Google searches and in Tweets. In addition to providing insight into the manifestations of these movements, which should help inform policymakers and community leaders who are attempting to reduce its occurrence and mitigate its harms, we also hope to examine whether these manifestations follow a temporal pattern which may provide additional insight. Specifically, we hypothesize that hate speech may first manifest through internet searches, as this is the most low-stakes, low-involvement form of participation. We believe participants may then move to expressing similar beliefs that they've seen in their searches on social media sites such as Twitter, which represents an escalation in their commitment to the beliefs. To do this, we need to build models to best predict if a tweet includes hate speech. We train, validate, and test multiple types of models and evaluate. We first tried more traditional supervised machine learning models like Logistic Regression, Bag of Words, and TF-IDF. Then we tested deep learning NLP models like bidirectional LSTM, Transformer distilBERT, and CNN. Once we find our best performing model, we hope to compare this to the Google Trends results to determine whether this temporal progression hypothesis seems plausible as a basis for future research.

Google Trends

Theoretical: What Are People Really Thinking? Use of Google Searches to Measure Hate Speech

In his book Everybody Lies, Seth Stephens-Davidowitz states, "Google serves as digital truth serum, as a modern-day confessional." Every time we type something into Google, we leave a trace of who we are and what we truly believe. In his paper referenced in the book, Racial Animus and Voting, he uses the percent of Google search queries that include racially charged language as a proxy for the area's racial animus. He chose to use Google data since it is unlikely to suffer from major social censoring due to the fact that Google searchers are online

and alone, which makes it easier to express socially taboo thoughts. People are not likely to admit to being racist in a professionally done survey, however their internet searches might show that they are. To measure this, Dr. Stephens-Davidowitz used variations of the n-word as his search term for states and major metropolitan areas in 2004-2007. Using this, he defined the racially charged search rate as the percentage of an area's searches including a racist word, taken from a random sample of total Google searches, divided by a common factor such that the top racist area has a value of 100. His work found that the racially charged search rate was a significant, negative predictor of Obama's 2008 and 2012 vote share, controlling for Kerry's 2004 vote share. This result is robust to controls like unemployment rate, demographic information, and changing liberalism in a region. He also adds in Google controls to reduce measurement error like the search rate for terms like "African Americans" which might suggest an area's interest in information related to African-Americans not racial animus. Also, as a check, he compared the results with a more traditional source of seeing racial prejudice, a survey about support for a law banning interracial marriage. While that survey seemed to perform similarly for Kerry and Obama's vote share suggesting there is a quiet group of people that voted for Kerry but not Obama due to racism. This makes sense since an Internet search is not an expressive act, as answering survey responses are, but a practical one intended to achieve a goal. The aim of answering a survey question is to express one's view, whereas the aim of a search is to find some information. Overall, the paper found that racial animus gave Obama's opponent the equivalent to a home-state advantage country-wide. Many additional studies have started to use Google searches as a proxy for an area's racism including *Association between an Internet-Based Measure of Area Racism and Black Mortality* by Chae, et al and *Income Inequality and White-on-Black Racial Bias in the United States: Evidence From Project Implicit and Google Trends* by Connor et al.

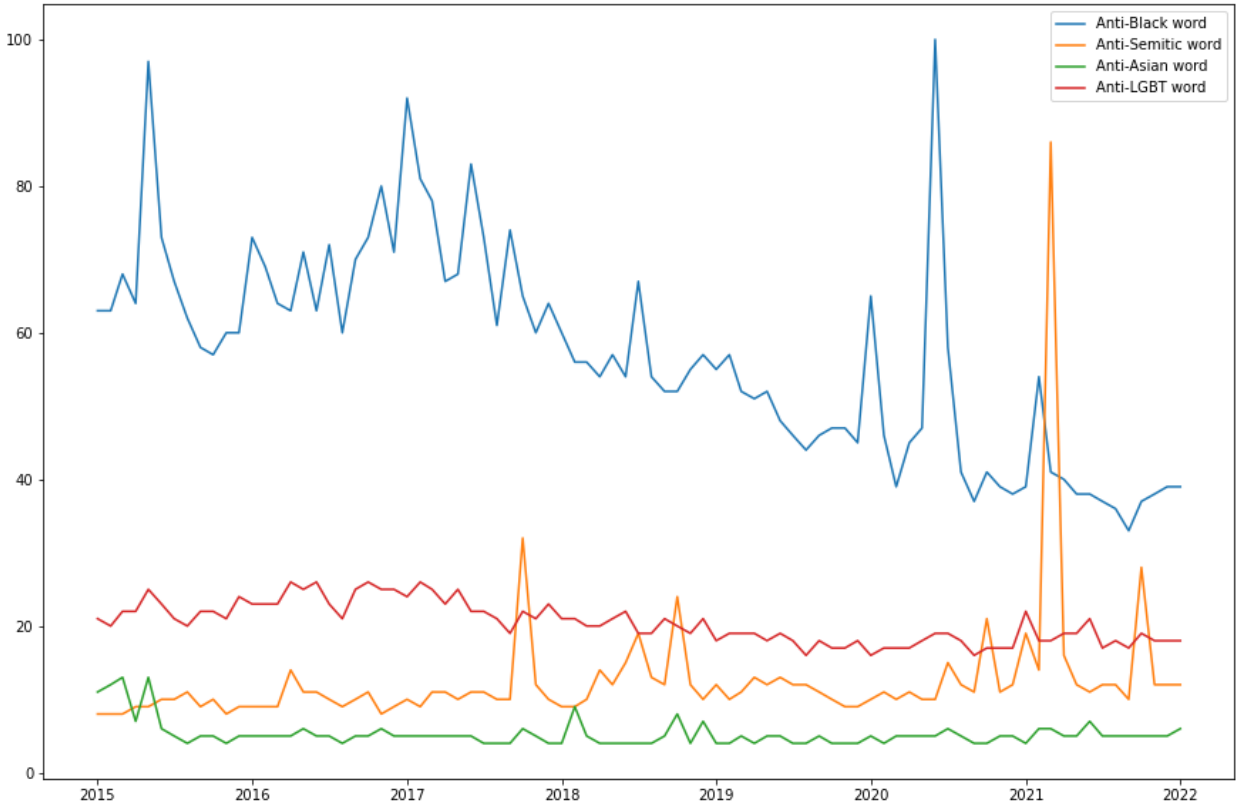
Another paper that uses Google Trends, along with Twitter, as a data source is Public Perceptions on Organised Crime, Mafia, and Terrorism: A Big Data Analysis based on Twitter and Google Trends by Kostakos. He explains that Google Trends content is passive while Twitter content is active. He used Twitter's API to collect tweets related to organized crime and used Zipf's Law to determine the internal consistency of the vocab distribution in the sample data. Similarly, he used Google Trends to collect Google searches surrounding the same topics used for the Twitter API. In conclusion, he found that both the Google search behavior and Twitter

behavior are feasible sources to draw insight from into the way crowds perceive social phenomena.

Based on the above research about Google searches in literature, it can be determined that there is precedence for using them as a proxy for people's true thoughts about a subject. Therefore, using Google searches as a measure of people's true feelings about hate speech is a worthwhile exercise and will add dynamism to our study on hate speech.

Application

To put this theory into practice, we used the Google Trends API to understand the temporal patterns of hate speech searches. We looked at the interest across the United States between 2015 and 2022 for various hate speech words. Since tweets do not generally have a geotab associated with them, we decided to stick with looking at hate speech Googles across the full United States. Specifically, we chose words that are not ever used in a context other than hate speech so that we did not conflate our results. As seen in the chart below, there are clear peaks in when hate speech is googled in the US. Additionally these trends differ depending on the minority group referred to.



Twitter

Twitter Theoretical: Hate Speech in Social Media

With wider usage of social media, determining best approaches to identify toxic languages becomes an expansive and layered problem to solve and many data scientists share a high level of interest to utilize machine learning and natural language processing to solve the identification quandary. For instance, the Conversation AI team, a research initiative founded by Jigsaw and Google (both a part of Alphabet) are working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion).[1] They have published two Kaggle challenges with unstructured online comment datasets with labeling informed by the Online Hate Index Research Project at D-Lab, University of California, Berkeley to train models and detect toxic and hate speech. (Toxic Comment Classification Challenge, Jigsaw Unintended Bias in Toxicity Classification)

In order to tackle this issue, firstly we must be able to define toxic language. We broadly divide toxic language into two categories: hate speech and offensive language.[2]Hate speech is defined as "language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate,or to insult the members of the group".[3] We define offensive language as the text which uses abusive slurs or derogatory terms.

Various machine learning algorithms have been developed to tackle the problem of toxic language in social media. Majority of the approaches deal with feature extraction from the text. One of the most popularly used text representation methods is the fixed-length vector space model of bag-of-words, due to its simplicity, efficiency and often surprisingly good level of accuracy and robustness. Lexical features perform well in detecting offensive entities, but it was observed that these features fail to understand the context of the sentences and ignore word semantics. They fail to distinguish sentences' offensiveness which contain the same words but in

[1] Jigsaw Unintended Bias in Toxicity Classification, https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification

[2] Watanabe, H., Bouazizi, M., & Ohtsuki, T. (2018). Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE access*, *6*, 13825-13835.

[3] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 11, No. 1, pp. 512-515).

different orders.[4] Linguistic features such as parts-of-speech are used in hate speech detection problems as well, consisting in detecting the category of the word, for instance, personal pronoun (PRP), Verb base forms (VB), Verb non-3rd person, Adjectives (JJ), etc.

Sentiment-based methods to detect abusive language are also prevalent in recent years' studies. Y. Chen, Y. Zhou, S. Zhu and H. Xu (2012) applied sentiment analysis to detect bullying in tweets and use Latent Dirichlet Allocation (LDA) topic models to identify relevant topics in these texts.

Deep learning techniques are being used in text classification and sentiment analysis using paragraph2vec approach. Convolutional Neural Network (CNN) based classification is being used in B. Gambck and U. Sikdar's study (2017), where they experimented with a system for Twitter hate-speech text classification based on a deep-learning CNN model.[5]

Badjatiya Pinkesh, et al (2017) work on deep learning for hate speech detection in tweets[6] gives us a good sample to begin with. As baseline methods, Badjatiya's team experimented with Char n-grams, TF-IDF and Bag of Words Vector(BoWV) to represent text. They investigated three neural network architectures after initializing the word embeddings with either random embeddings or GloVe embeddings: 1) Use CNNs for hate speech detection; 2) Utilize LSTM to process arbitrary sequences of inputs and capture long range dependencies in tweets; 3) FastText not only represents a document by average of word vectors similar with BoWV model but also allows update of word vectors through Back-propagation during training and fine-tune the word representation according to the task. For each network, they are trained and fine-tuned using labeled data with back-propagation. They also experimented using network methods to learn task-specific word embeddings tuned towards the hate speech labels and using these embeddings as features and various other classifiers like SVMs and GBDTs as the learning method. Their result showed that embeddings learned from deep neural network models when combined with gradient boosted decision trees led to best recall values of 0.93.

---

[4] Liu, S., & Forss, T. (2015, November). New classification models for detecting Hate and Violence web content. In 2015 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K) (Vol. 1, pp. 487-495). IEEE.

[5] Gambäck, B., & Sikdar, U. K. (2017, August). Using convolutional neural networks to classify hate-speech. In Proceedings of the first workshop on abusive language online (pp. 85-90).

[6] Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In Proceedings of the 26th international conference on World Wide Web companion (pp. 759-760).

Getting the Data

The dataset we used for training and testing models is from Twitter Hate Speech Kaggle dataset. Every tweet was labeled 1 if the author classified it as hate speech and 0 as non-hate speech. The training dataset has 29,530 data entries (29,720 labeled as hate speech and 2,242 labeled as non-hate speech). The testing data set contains 16,130 data entries but without labels.

The other dataset we used for model testing and determining similarities with Google Search Trends is from Waseem, Z., & Hovy, D. (2016) work.[7] They annotated 16,914 tweets in 2015, 3,383 of that for sexist content sent by 613 users, 1,972 for racist content sent by 9 users, and 11,559 for nei- ther sexist or racist and is sent by 614 users. Since the dataset is only available as tweet IDs and labels at their Github repository, we used Snscrape package to scrape tweets and its date info based on tweet IDs.[8]

The other potentially useful datasets for future training and testing are as followings: the Conversation AI team, a research initiative founded by Jigsaw and Google have published two Kaggle challenges with unstructured online comment datasets with labeling informed by the Online Hate Index Research Project at D-Lab, University of California, Berkeley to train models and detect toxic and hate speech. (Toxic Comment Classification Challenge, Jigsaw Unintended Bias in Toxicity Classification); Barbieri, F., Camacho-Collados, J., Neves, L., & Espinosa-Anke, L. (2020) published TweetEval Dataset consisting of seven heterogeneous Twitter-specific classification tasks (Emoji Prediction, Emotion Recognition, Hate Speech Detection, Irony Detection, Offensive Language Identification, Sentiment Analysis, Stance Detection).

Cleaning and Preprocessing the Data

Before being able to run our models, we needed to take some steps to clean the data and get it into an analyzable form. Since we have a Twitter dataset, we would like to remove some substrings commonly used on the platform like the hashtags #, retweet marks, user mentions @ and hyperlinks. We'll use the re library to perform regular expression operations on our tweet. We'll define our search pattern and use the sub() method to remove matches by substituting with

[7] Waseem, Z., & Hovy, D. (2016, June). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL student research workshop (pp. 88-93).
[8] Snscrape package source code: https://github.com/JustAnotherArchivist/snscrape/blob/master/snscrape/modules/twitter.py

an empty character (i.e. ''). We also remove stopwords based on nltk.corpus package and remove punctuations. Then we tokenize the text using TweetTokenizer from nltk.tokenize package. The following is an example of our text preprocessing results.

Original Text: @user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run

NLTK Twitter tokenized: @user father dysfunctional selfish drags kids dysfunction run

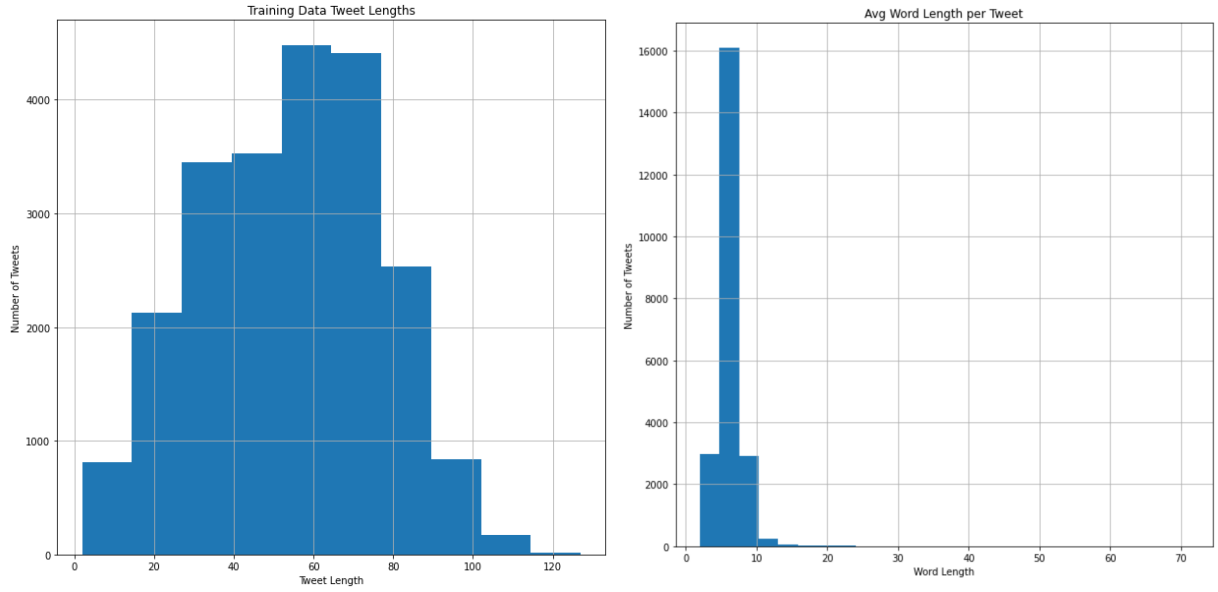
After final cleaning: father dysfunctional selfish drags kids dysfunction run

Original vs. Upsampled Data

Given the severe imbalance in our data between hateful and non-hate tweets, we created a secondary datasource by upsampling the original data. In this upsampling, we continuously resampled with replacement from the hateful tweets until the hateful and non-hateful tweets were of equal proportion. In all of our analyses, we tested classification performance on both the original and upsampled datasets, and we found consistently better performance with the upsampled data.

Data Exploration

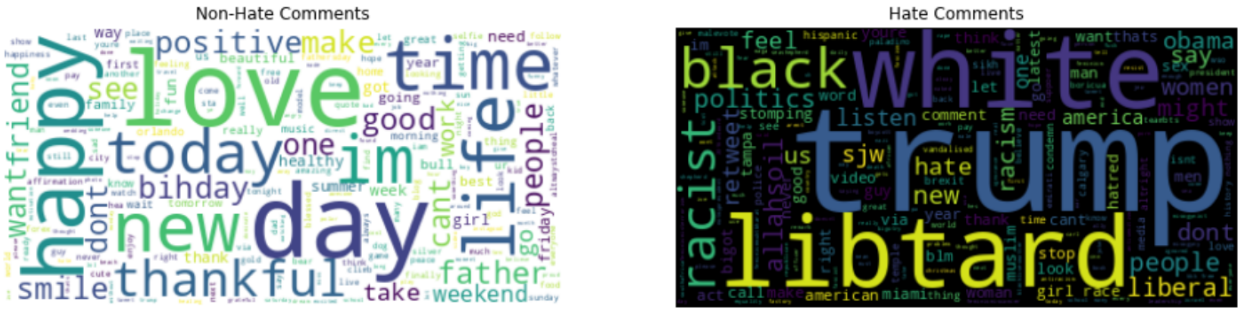
After cleaning our data and creating an additional upsampled dataset, we performed initial exploratory analysis of the training data. Below we show the distribution of tweet lengths in our training dataset and see that tweets are roughly normally distributed in length, having a mean length of 54 and a slightly larger amount of shorter tweets than longer tweets. When we consider the average word length per tweet, it is less normally distributed, however: the majority of words are less than 10 characters in length, while a very few extend to much longer lengths.



Distribution of tweet lengths in training data (left) and average word length per tweet in training data (right).

We also considered word clouds of our hateful and non-hateful training data to examine the words likely to impact our model. To ensure our upsampled data was similar to our original data, we created further word clouds of the upsampled data and confirmed that the upsampled and original hate comments had similar contents, as did the non-hate comments.

Word clouds of Hate and Non-Hate comments in the original training data.



Word clouds of Hate and Non-Hate comments in the upsampled training data.

Evaluation

We decide to emphasize on F1 score and recall score to minimize false negatives when determining the best hyper-tuned models since our priority is detecting more potentially toxic comments rather than being over-cautious in not mislabeling non-toxic tweets as toxic. Since we are dealing with skewed datasets, where the percentage of toxic comments is the minority class, accuracy is not the preferred performance measure. Instead, throughout our model evaluations below, we will prefer Recall and F1 scores. The recall score tells us what proportion of toxic comments are correctly labeled, the precision score tells us what the proportion of positive-labeled comments are actual toxic comments, and the F1 score is the weighted average of Precision and Recall.

Models

We first began training classifiers with traditional supervised machine learning methods to develop a baseline for classification performance. Because of its simplicity, we first employed manually-implemented Bag-of-Words features, both binary and count for both unigram and bigram vocabularies, in logistic regression models as our baseline. We then developed a full suite of traditional models using scikit-learn's implementation of a count-based BoW and TF-IDF. We employed Logistic regression, multinomial Naive Bayes, Support Vector Classification, Linear Support Vector Classification, Gradient Boosting Classifier, XGBoost Classifier, Random Forest Classifier, and KNeighborsClassifier to thoroughly test performance on traditional methods.

We next tested classification performance on multiple deep learning algorithms, including bi-directional Long Short Term Memory networks (LSTM), Bidirectional Encoder

Representations from Transformers (BERT), and Convolutional Neural Network (CNN), to evaluate whether deep learning models improve the classification accuracy over our traditional models.

Traditional Supervised ML Models

For our initial baseline, we generated our own vocabularies and feature sets to employ in a scikit-learn logistic regression model. As it was unclear which feature structure would be optimal for Twitter data for this purpose, we generated four sets of features: (1) unigram binary, which utilizes each distinct word as a token and contains a single 1 or 0 indicating the presence or absence of the token in the sequence; (2) unigram count, which utilizes each distinct word as a token and contains a count of the number of times the token occurs in the sequence, (3) bigram binary, which utilizes bigrams as the tokens and contains a single 1 or 0 indicating the presence or absence of the token in the sequence; and (4) bigram count, which utilizes bigrams as the tokens and contains a count of the number of times the token occurs in the sequence.

After generating vocabularies from our training data for each of these feature types, and then generating features alongside their corresponding labels for the training, validation, and test data, we generated multiple logistic regression models to test the performance.

Logistic regression with scikit learn, original dataset

Metric	**Unigram Binary**	**Unigram Count**	**Bigram Binary**	**Bigram Count**
Validation Accuracy	0.9559	0.9547	0.9282	0.9282
Validation Binary Recall	0.5145	0.5116	0.0000	0.0000
Validation Macro Recall	0.7523	0.7503	0.5000	0.5000
Validation Macro F1	0.8016	0.7973	0.4814	0.4814
Test Accuracy	0.9570	0.9558	0.9330	0.9330
Test Binary	0.5296	0.5265	0.0000	0.0000

Recall				
Test Macro Recall	0.7586	0.7565	0.5000	0.5000
Test Macro F1	0.8000	0.7955	0.4827	0.4827

Given the somewhat surprising success of the unigram binary features when considering this initial set of basic, logistic regression models, we further examined which tokens contributed most to the unigram binary logistic regression model's performance. We performed an analysis of the weights which most and least contributed to a designation of a tweet as hateful or not hateful:

Most hateful words:
4.2309 | allahsoil
3.1589 | racism
2.6945 | misogyny
2.6012 | bigot
2.5492 | 2017
2.5025 | white
2.2742 | latest
2.1515 | racist
2.1493 | treason
2.1267 | misogynist

Least hateful words:
-3.0948 | bihday
-2.6052 | day
-2.5491 | orlando
-2.3904 | positive
-2.3258 | healthy
-2.2211 | friday
-2.1652 | weekend
-2.1620 | hardcore
-2.1542 | thankful
-2.1484 | days

Most neutral words:
0.0000 | <pad>
0.0000 | remains
0.0001 | grandad
0.0001 | awasome
0.0001 | inwoo

0.0002 | selfrespect
0.0003 | titanic
0.0003 | lauren
0.0004 | ebay
0.0005 | systems

We next performed the same analysis with an upsampled dataset, and we found the performance improved.

Logistic regression with scikit learn, upsampled dataset

Metric	**Unigram Binary**	**Unigram Count**	**Bigram Binary**	**Bigram Count**
Validation Accuracy	0.9650	0.9641	0.4971	0.4971
Validation Binary Recall	0.9843	0.9847	0.9966	0.9966
Validation Macro Recall	0.9652	0.9644	0.5040	0.5040
Validation Macro F1	0.9650	0.9641	0.3421	0.3421
Test Accuracy	0.9631	0.9630	0.5035	0.5035
Test Binary Recall	0.9854	0.9868	0.9951	0.9951
Test Macro Recall	0.9630	0.9629	0.5028	0.5028
Test Macro F1	0.9631	0.9629	0.3441	0.3441

With the upsampled data, we had to increase the maximum number of iterations allowed in our model to 200 to allow for full convergence. We do see much better performance with the upsampled data than we did with the original dataset, particularly for the unigram vocabularies, though again somewhat surprisingly the unigram binary vocabulary produced the best results, if only slightly. An examination of the most impactful words for the model, compared to the model run using the original, not unsampled data, finds that while many of the same words contribute

heavily (for example, “allahsoil”, “racism”, “bigot” among the most hateful), there are also some noteable differences which may reflect which hateful tweets were included in the upsampling (for example, “misogyny” and “treason” are no longer among the most hateful, though “equality” and “blacklivesmatter” are now included).

Most hateful words:

5.9076 | allahsoil
4.2857 | racism
4.0006 | bigot
3.4879 | equality
3.4789 | white
3.3869 | blacklivesmatter
3.3394 | neighbors
3.2137 | mc
3.2048 | blatantly
3.2015 | shitty

Least hateful words:

-3.9929 | bihday
-3.2502 | orlando
-3.1340 | hardcore
-2.8810 | thankful
-2.7886 | healthy
-2.7294 | friday
-2.7153 | getting
-2.6452 | day
-2.6402 | tomorrow
-2.5961 | followers

Most neutral words:

0.0000 | <pad>
0.0000 | elder
0.0000 | litter
0.0000 | vaccines
0.0000 | devoted
0.0000 | rooting
0.0000 | minime
0.0000 | mornin
0.0000 | ankara
0.0000 | pl

After this initial performance assessment using manually-developed vocabularies and features, we next performed an extensive assessment of many common traditional unsupervised machine learning methods using scikit-learn's implementation of count-based BoW and TF-IDF. The models we included in this full comparison were: Logistic regression, multinomial Naive Bayes, Support Vector Classification, Linear Support Vector Classification, Gradient Boosting Classifier, XGBoost Classifier, and Random Forest Classifier, KNeighborsClassifier. As expected, models trained on more balanced datasets tended to perform better in terms of our key metrics, recall and F1 scores. The models performance is as follows:

Bag-of-words Based Models (Original dataset)

	Validation Accuracy	**Validation Binary Recall**	**Validation Macro Recall**	**Validation Binary F1**	**Validation Macro F1**	**Test Accuracy**	**Test Binary Recall**	**Test Macro Recall**	**Test Binary F1**	**Test Macro F1**
LogisticRegression	0.9608	0.9105	0.9367	0.6479	0.8136	0.9633	0.8962	0.9311	0.6508	0.8157
MultinomialNB	0.9582	0.8673	0.9147	0.6296	0.8038	0.9599	0.8177	0.8920	0.6336	0.8062
SVC	0.9551	0.9329	0.9444	0.5639	0.7701	0.9595	0.9320	0.9462	0.5855	0.7821
LinearSVC	0.9461	0.9574	0.9517	0.4110	0.6914	0.9522	0.9340	0.9433	0.4637	0.7193
GradientBoostingClassifier	0.9451	0.9175	0.9316	0.4036	0.6874	0.9497	0.8922	0.9216	0.4303	0.7020
XGBClassifier	0.9499	0.8824	0.9171	0.5000	0.7368	0.9560	0.8667	0.9128	0.5520	0.7644
RandomForestClassifier	0.9570	0.8416	0.9018	0.6227	0.8000	0.9626	0.8586	0.9129	0.6551	0.8177
KNeighborsC	0.9378	0.8485	0.8938	0.2732	0.6203	0.9420	0.8116	0.8777	0.2872	0.6285

lassifier										

Bag-of-words Based Models (Upsampled dataset)

	Validation Accuracy	Validation Binary Recall	Validation Macro Recall	Validation Binary F1	Validation Macro F1	Test Accuracy	Test Binary Recall	Test Macro Recall	Test Binary F1	Test Macro F1
LogisticRegression	0.9484	0.6336	0.8039	0.6506	0.8114	0.9526	0.6298	0.8044	0.6676	0.8211
MultinomialNB	0.9253	0.4878	0.7362	0.6092	0.7839	0.9217	0.4536	0.7200	0.5847	0.7708
SVC	0.9614	0.8664	0.9161	0.6702	0.8249	0.9631	0.8396	0.9042	0.6679	0.8242
LinearSVC	0.9401	0.5643	0.7714	0.6353	0.8013	0.9407	0.5414	0.7616	0.6302	0.7990
GradientBoostingClassifier	0.9328	0.5316	0.7479	0.5347	0.7492	0.9366	0.5228	0.7471	0.5620	0.7639
XGBClassifier	0.9328	0.5243	0.7498	0.5955	0.7794	0.9372	0.5226	0.7510	0.6055	0.7857
RandomForestClassifier	0.9534	0.7216	0.8445	0.6386	0.8068	0.9604	0.7510	0.8617	0.6735	0.8262
KNeighborsClassifier	0.9015	0.3590	0.6586	0.4085	0.6774	0.9013	0.3397	0.6513	0.4050	0.6756

TF-IDF Based Models (Only Unigram, original dataset)

	Validation Accuracy	Validation Binary Recall	Validation Macro Recall	Validation Binary F1	Validation Macro F1	Test Accuracy	Test Binary Recall	Test Macro Recall	Test Binary F1	Test Macro F1
LogisticRegression	0.9476	0.9604	0.9539	0.4360	0.7042	0.9503	0.9278	0.9393	0.4306	0.7023
MultinomialNB	0.9395	0.9821	0.9605	0.2750	0.6217	0.9434	1.0000	0.9714	0.2695	0.6201
SVC	0.9570	0.9539	0.9555	0.5847	0.7810	0.9599	0.9329	0.9468	0.5915	0.7852
LinearSVC	0.9301	1.0000	0.9650	0.0510	0.5073	0.9359	1.0000	0.9679	0.0836	0.5252
GradientBoostingClassifier	0.9438	0.8713	0.9083	0.3955	0.6830	0.9491	0.8667	0.9088	0.4272	0.7003
XGBClassifier	0.9493	0.8976	0.9242	0.4841	0.7287	0.9570	0.8912	0.9251	0.5598	0.7686
RandomForestClassifier	0.9597	0.8683	0.9160	0.6485	0.8135	0.9637	0.8657	0.9168	0.6667	0.8237
KNeighborsClassifier	0.9457	0.8333	0.8910	0.4468	0.7091	0.9522	0.8651	0.9098	0.4877	0.7313

TF-IDF Based Models (Only Bigram, original dataset)

	Validation Accuracy	Validation Binary Recall	Validation Macro Recall	Validation Binary F1	Validation Macro F1	Test Accuracy	Test Binary Recall	Test Macro Recall	Test Binary F1	Test Macro F1
LogisticRegression	0.9357	0.9737	0.9545	0.1937	0.5801	0.9403	0.9730	0.9565	0.2011	0.5851

MultinomialNB	0.9363	1.0000	0.9679	0.2037	0.5852	0.9414	1.0000	0.9704	0.2216	0.5956
SVC	0.9411	0.9844	0.9625	0.3088	0.6390	0.9453	0.9836	0.9642	0.3141	0.6428
LinearSVC	0.9301	1.0000	0.9650	0.0510	0.5073	0.9359	1.0000	0.9679	0.0836	0.5252
GradientBoostingClassifier	0.9359	0.9744	0.9550	0.1984	0.5825	0.9412	0.9535	0.9473	0.2253	0.5973
XGBClassifier	0.9359	0.9744	0.9550	0.1984	0.5825	0.9416	1.0000	0.9705	0.2265	0.5981
RandomForestClassifier	0.9489	0.9806	0.9644	0.4519	0.7125	0.9541	0.9810	0.9672	0.4836	0.7298
KNeighborsClassifier	0.9363	1.0000	0.9679	0.2037	0.5852	0.9416	1.0000	0.9705	0.2265	0.5981

TF-IDF Based Models (Only Unigram, upsampled dataset)

	Validation Accuracy	**Validation Binary Recall**	**Validation Macro Recall**	**Validation Binary F1**	**Validation Macro F1**	**Test Accuracy**	**Test Binary Recall**	**Test Macro Recall**	**Test Binary F1**	**Test Macro F1**
LogisticRegression	0.9453	0.6000	0.7888	0.6525	0.8114	0.9460	0.5711	0.7773	0.6579	0.8143
MultinomialNB	0.9140	0.4470	0.7166	0.5822	0.7671	0.9090	0.4117	0.6994	0.5514	0.7504
SVC	0.9624	0.9409	0.9521	0.6604	0.8202	0.9654	0.9235	0.9453	0.6706	0.8262
LinearSVC	0.9273	0.4962	0.7387	0.6027	0.7814	0.9278	0.4770	0.7312	0.5995	0.7799
Gradie	0.9317	0.5238	0.7442	0.5335	0.7483	0.9355	0.5162	0.7434	0.5528	0.7590

ntBoostingClassifier										
XGBClassifier	0.9263	0.4905	0.7323	0.5679	0.7638	0.9270	0.4700	0.7241	0.5647	0.7624
RandomForestClassifier	0.9620	0.8375	0.9030	0.6884	0.8341	0.9645	0.8186	0.8953	0.6953	0.8383
KNeighborsClassifier	0.9117	0.4192	0.6935	0.4922	0.7219	0.9182	0.4256	0.6991	0.5088	0.7321

TF-IDF Based Models (Only Bigram, upsampled dataset)

	Validation Accuracy	**Validation Binary Recall**	**Validation Macro Recall**	**Validation Binary F1**	**Validation Macro F1**	**Test Accuracy**	**Test Binary Recall**	**Test Macro Recall**	**Test Binary F1**	**Test Macro F1**
LogisticRegression	0.9357	0.9737	0.9545	0.1937	0.5801	0.9403	0.9730	0.9565	0.2011	0.5851
MultinomialNB	0.9363	1.0000	0.9679	0.2037	0.5852	0.9414	1.0000	0.9704	0.2216	0.5956
SVC	0.9411	0.9844	0.9625	0.3088	0.6390	0.9453	0.9836	0.9642	0.3141	0.6428
LinearSVC	0.9301	1.0000	0.9650	0.0510	0.5073	0.9359	1.0000	0.9679	0.0836	0.5252
GradientBoostingClassifier	0.9359	0.9744	0.9550	0.1984	0.5825	0.9412	0.9535	0.9473	0.2253	0.5973
XGBClassifier	0.9359	0.9744	0.9550	0.1984	0.5825	0.9416	1.0000	0.9705	0.2265	0.5981
RandomFore	0.9489	0.9806	0.9644	0.4519	0.7125	0.9541	0.9810	0.9672	0.4836	0.7298

stClassifier										
KNeighborsClassifier	0.9363	1.0000	0.9679	0.2037	0.5852	0.9416	1.0000	0.9705	0.2265	0.5981

Overall, we see the models using BoW perform better with the original, imbalanced dataset than the upsampled, but when using TF-IDF, the upsampled dataset performed better.

Deep Learning Models

LSTM

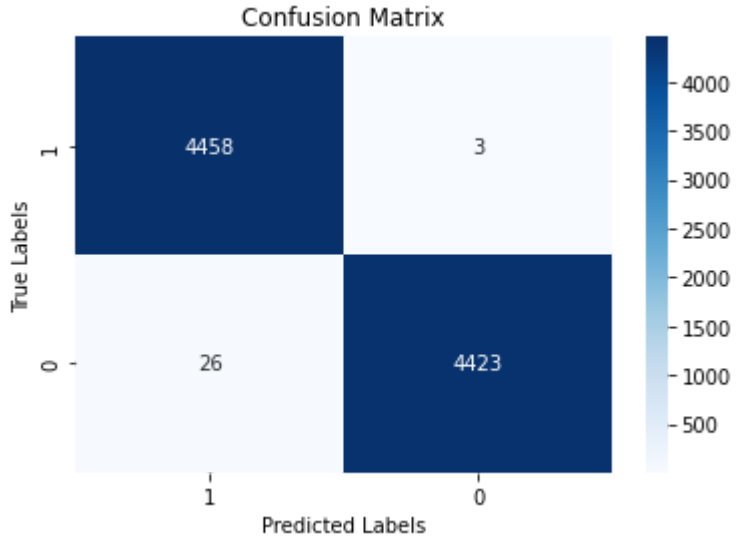
We next moved onto deep learning methods, using an bi-LSTM model to classify whether a tweet contained hate speech. This was a good choice for many reasons. For one, LSTM allows us to deal with words that are out of the vocabulary with the unknown tag. This means our model is still able to run even if new words are introduced in the validate or test dataset. Also, while LSTM models are not able to handle sequences of different lengths, there are helpful features that will pad the sequence to make all tweets in the batch the same length. However, since we don't want to actually use this padded token to decide whether the tweet has hate speech in it, we need to then pack the padding so that it is not shown to our model. In bidirectional LSTM, instead of training a single model, we introduce two. The first model learns the sequence of the input provided, and the second model learns the reverse of that sequence. Since we do have two models trained, we need to build a mechanism to combine both. In our model, we concatenate the forward sequence and the reverse sequence together to use in our dropout and linear layers.

We decided to use the Glove vocabulary to have a constant set of vocab words to compare our tokenized tweets to. Specifically, we used the Glove twitter vocabulary since that was created using tweets and includes internet shorthands for words that are more common in social media posts than in other forms of writing. Our model brought all this together by including an embedding layer, packed sequence, LSTM, pad packed sequence, a dropout layer, a linear layer, and then finally a Sigmoid activation function. Sigmoid makes sense since we are interested in predicting the probability whether the tweet has hate speech or does not have hate

speech so a 0-1 prediction makes sense. Once we have our model and defined our optimizer, we can train our model. We chose to train it over 20 epochs - and inside the training function, we validate as we go by calculating the Binary Cross Entropy loss. Once we found our best model, we ran the training data through that model to evaluate the model.

bi-LSTM Model Results

	Precision	Recall	F1 Score	Support
1	0.9942	0.9993	0.9968	4461
0	0.9993	0.9942	0.9967	4449
Accuracy			0.9967	8910
Macro average	0.9968	0.9967	0.9967	8910
Weighted average	0.9968	0.9967	0.9967	8910



Transformer Model: distilBERT

We further attempted to develop a classifier using the distilBERT model, which is based on the full-sized BERT model developed by Devlin et al. (2018), which is in turn based on the Transformer architecture of Vaswani et al. (2017). BERT is a pre-trained language model which increased performance substantially on language-based tasks over previous models when it came

out and has been widely used since its release. distilBERT was developed as a smaller, less computationally expensive alternative to allow users to maintain much of the performance of BERT without requiring the investment of time and resources which BERT requires. Sanh et al. (2019) utilized knowledge distillation, also called teacher-student training, to train distilBERT to mimic the performance of the full-sized BERT model. Using this technique, the authors were able to "reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster" (Sanh et al. 2019). We found that using distilBERT and the upsampled data, we were able to obtain very high performance, almost matching the bi-LSTM, with a test recall of 0.991.

distilBERT Model Results

	Original, non-upsampled data	Upsampled data
Test Accuracy	0.965989	0.991358
Test F1	0.965985	0.991358
Test Loss	0.151185	0.032449
Test Recall	0.965985	0.991358

CNN

Convolutional Neural Networks (CNNs) are usually implemented in image recognition and classification since it is designed to identify indicative local predictors in a large structure, and to combine them to produce a fixed size vector representation of the structure, capturing the local aspects that are most informative for the prediction task at hand.[9] Since text data is a kind of sequence data, while time sequentiality of the general text data is relatively weak, text classification is usually less relevant to the sequential structure of the full text. Therefore, CNN-based text classification has gradually become a research hotspot when dealing with issues

[9] Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, *10*(1), 1-309.

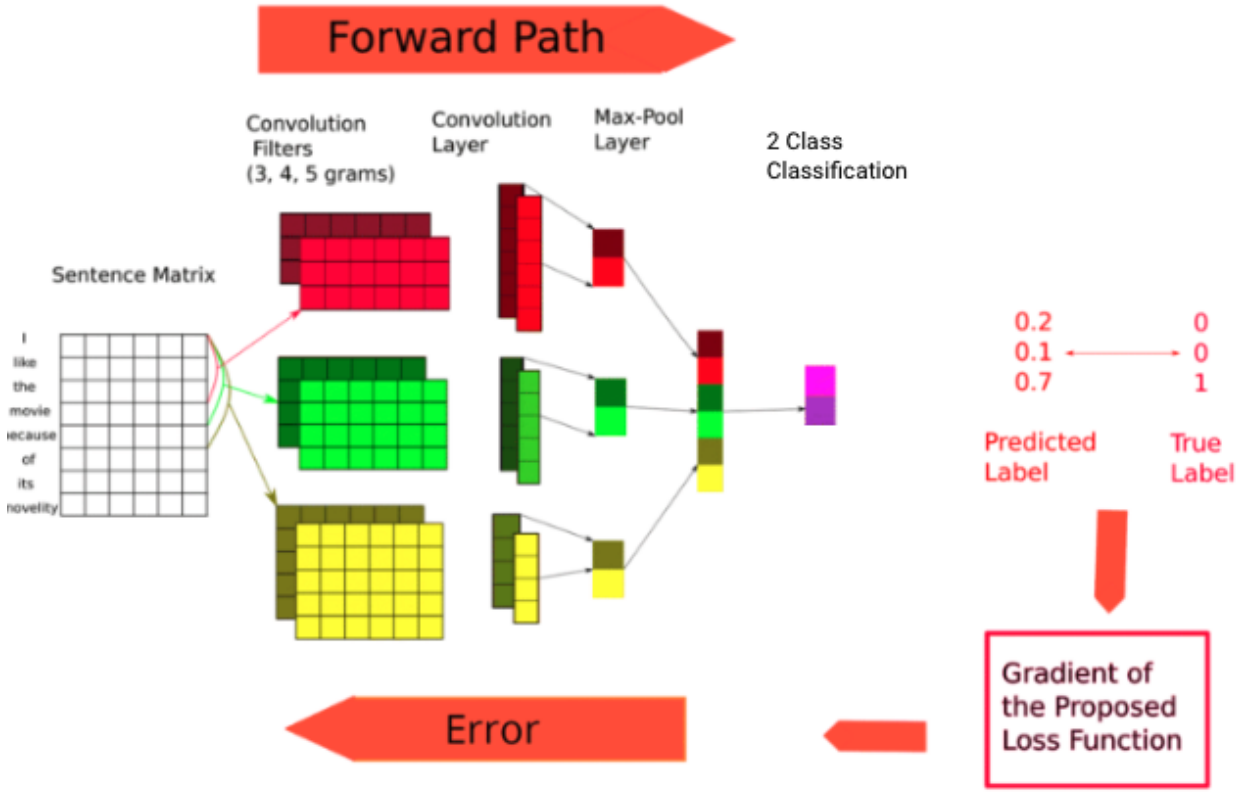
of text classification.[10] The CNN can extract the features of the text data accurately and reduce the complexity of models at the same time.[11]

The parameters used for CNN are: Filter sizes: 3, 4, 5; Drop out rate = 0.5 and Max-Pool layer to avoid overfitting; Learning Rate = 0.001, Batch size = 20, Binary entropy loss. Since the convolutional architecture will identify ngrams that are predictive for the task at hand, without the need to pre-specify an embedding vector for each possible ngram, we ran the model both using Glove embedding and using random embedding, which showed no much difference in model performance.

The model performance shows as follows. Though the weighted average recall score and f1-score are both 0.95, while for the class labeled as 1, the recall score is only 0.54 and the f1-score is 0.61. The CNN model does not show improvement when training on the upsampled dataset.

Compared to the other two models, the classification performance of CNN is relatively low. Given the fact that CNN is in essence a feature-extracting architecture, CNN could be instead integrated into a larger network and trained to work in tandem. For instance, we could use CNN layers to extract meaningful text sub-structures that are useful for classification tasks in future work.

[10] Ce, P., & Tie, B. (2020). An Analysis Method for Interpretability of CNN Text Classification Model. *Future Internet*, *12*(12), 228.
[11] Wang, Z., & Qu, Z. (2017, October). Research on Web text classification algorithm based on improved CNN and SVM. In 2017 IEEE 17th International Conference on Communication Technology (ICCT) (pp. 1958-1961). IEEE.

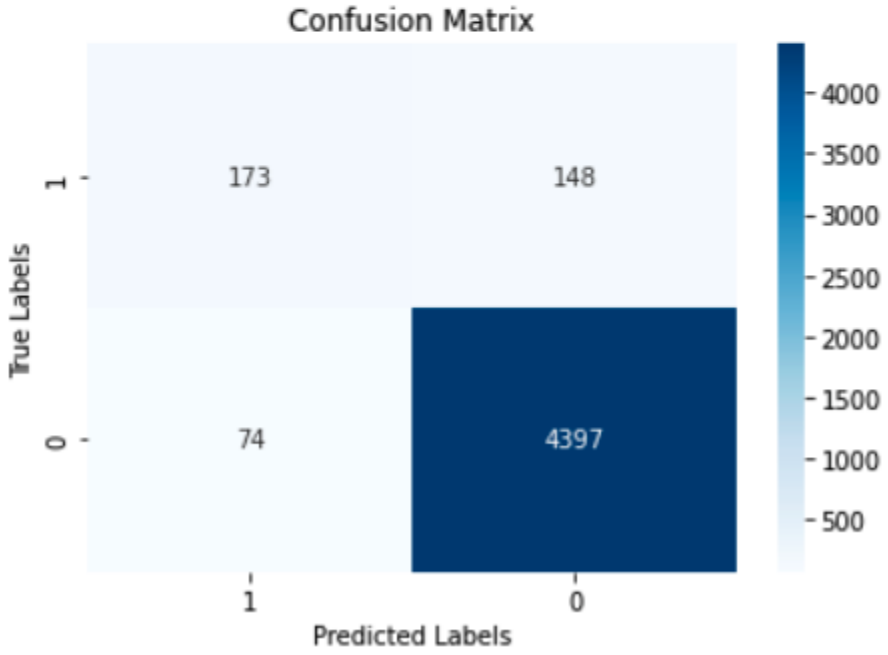


CNN Structure[12]

CNN Model Results - Original

	Precision	Recall	F1 Score	Support
1	0.7004	0.5389	0.6092	321
0	0.9674	0.9834	0.9754	4471
Accuracy			0.9537	4792
Macro average	0.8339	0.7612	0.7923	4792
Weighted average	0.9495	0.9537	0.9508	4792

[12] Graphic is based on Hajiabadi, H., Molla-Aliod, D., Monsefi, R., & Yazdi, H. S. (2020). Combination of loss functions for deep text classification. *International Journal of Machine Learning and Cybernetics*, *11*(4), 751-761.



Based on the above, the model that performed best was the LSTM model since that model had the highest recall and F1 Score.

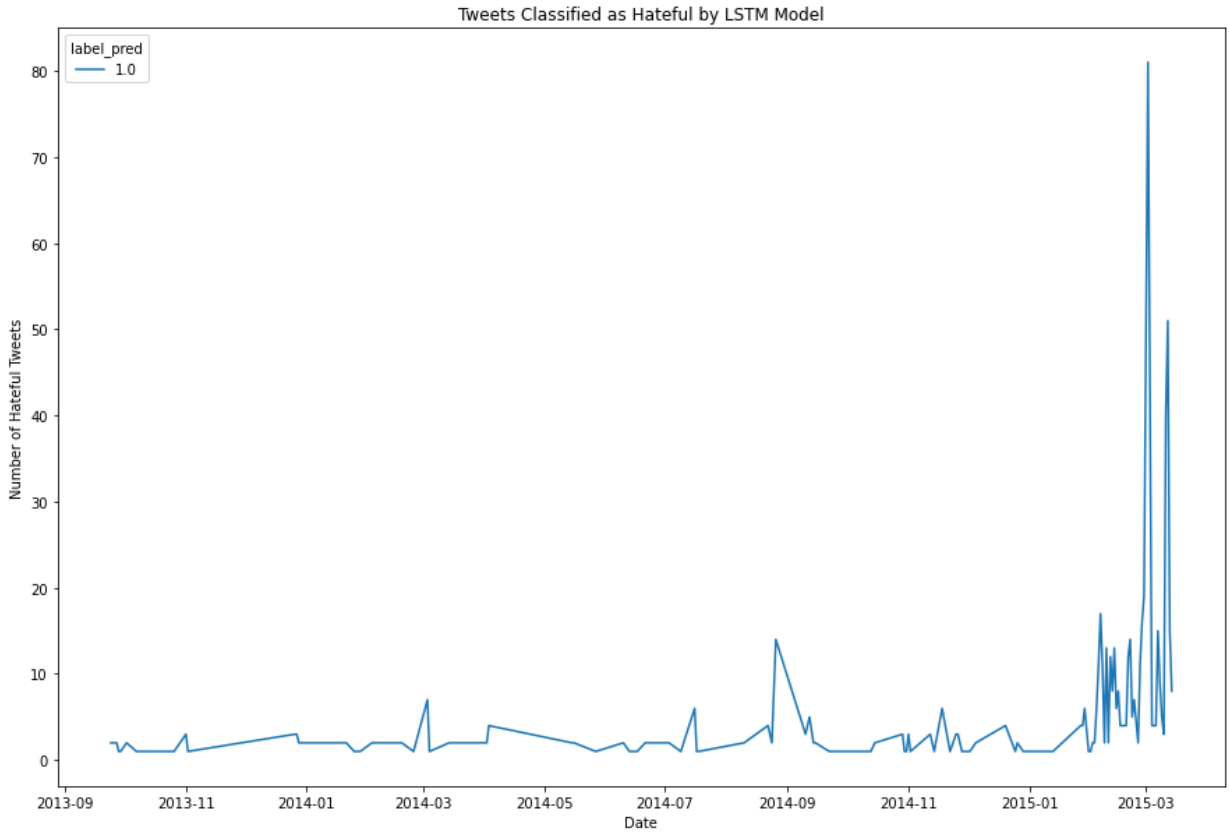
Testing our Predictions

We next used our best-performing model, the bi-LSTM, to predict which tweets have hate speech from the Waseem, Z., & Hovy, D. (2016) twitter corpus. While using Snscrape to obtain the dates for each tweet in this dataset, we were unable to collect this supporting information for many of the tweets. This left us with a smaller and potentially biased dataset, as we are not sure whether the tweets which we weren't able to obtain dates for.

This dataset includes its own labels identifying racist and sexist tweets, and it initially seemed that our model struggled to capture many of the tweets labeled as either racist or sexist: our model labeled 411 tweets as hateful which were labeled as not hateful in the dataset, our model did not label as hateful 2497 tweets identified as sexist, and our model did not label as hateful 15 tweets identified as racist. When we plot the tweets classified as hate speech over time, we also see an initially suspicious, sharp spike in hateful tweets in March 2015.

Comparative performance of our LSTM predictions with labeled tags

Dataset Labeled Tag	LSTM Predicted Label	Number of Tweets
none	0	12417
none	1	411
racism	0	15
sexism	0	2497
sexism	1	213



Temporal distribution of predicted hate speech on Waseem and Hovy Twitter dataset.

However, upon closer examination of the tweets for which our model differed from their labels, we find our model actually performed relatively close to our expectations. Many of the tweets which were labeled as hate speech by our model but not labeled as either racism or sexism

in the Waseem and Hovy data were expressing disdain and using hateful comments, just not directed at a particular race or gender. This overlapped with the spike in hate tweets in March 2015, as it appears many of those which our model labeled hateful but which were labeled 'none' by Waseem and Hovy were related to an Australian cooking show, My Kitchen Rules, which was slammed as promoting bullying and bad behavior in 2015 amid calls for a boycott.[13] The spike in hate tweets in March 2015 were related to the new season of this show, with comments like:

```
"#MKR This is fast becoming like that crap 80's movie- Neverending Story! More instant restaurants- Aaaaargh!"
```

```
'@mykitchenrules more rigged than the block. This will be the last time I tweet about and even watch #mkr \nHave fun with your ratings fall'
```

While these are not racism or sexism, and thus we would not expect them to be labeled as such in the Waseem and Hovy data, they are expressing very negative sentiment and thus we would expect our LSTM model, which was trained on more general hateful comments, to classify this as hate speech.

Similarly, when we consider the tweets which were not classified as hate speech by our model but which were considered sexism by Waseem and Hovy, we find that, while the tweets are derogatory towards women, many of them are not as outwardly hostile as the hate speech in the twitter corpus we used for training. For example, some of the tweets classified as sexism but not considered hate speech by our LSTM model were:

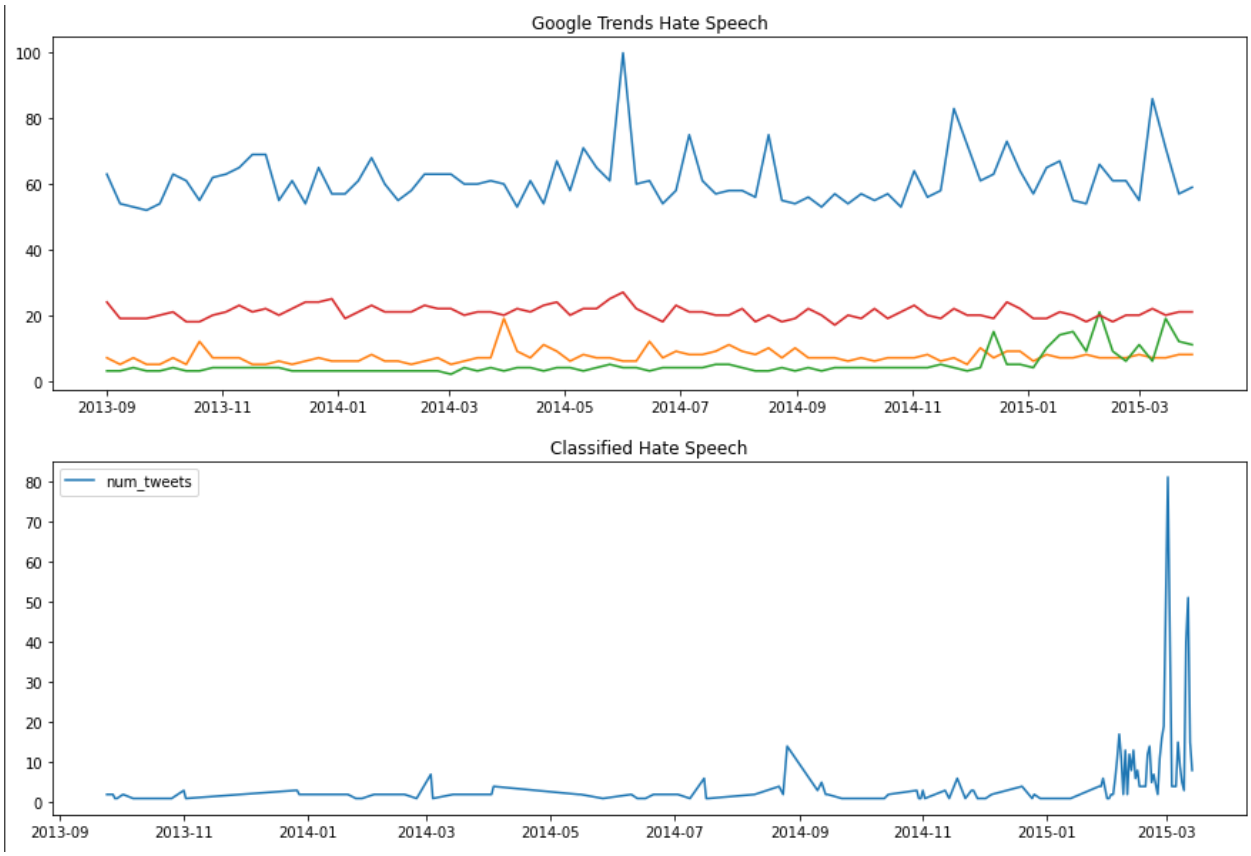
```
'Ash found her inner bogun #whistle #fingersinthemouth Ewwww #MKR'
```

```
'@mykitchenrules Elegant and beautiful?Cheap and trashy!Nothing more unattractive than girls banging on about how hot hey are. #mkr #notsassy'
```

While the second we might have hoped our model would identify as hate speech, it is clear from these examples that the sexism is more muted than the kind of severe hate speech in our training data, and thus our model can likely be excused for these classifications.

[13] MKR 2015: Calls for viewers to boycott My Kitchen Rules 2015 | news.com.au — Australia's leading news site

Next, we considered if there were any temporal patterns of hate tweets and how they lined up with the original Google Trends temporal patterns, returning to our original hypothesis that we might see spikes in Google prior to spikes in Twitter of similar hateful speech.



Google Trends identified hate speech and LSTM classified hate speech, 09/2013 - 03/2015

Unfortunately, our temporal comparison does not display any particularly strong mirrored trends, aside from perhaps the spike in Google Trends hate speech before September 2014 which was immediately followed by a spike in our classified hate speech. Waseem and Hovy did only utilize a relatively small subset of all tweets in their dataset, however, so for a more thorough analysis, a future study should predict using significantly more tweets to see if any stronger patterns emerge.

Lessons Learned & Limitations

Learnings include gaining hands-on experience with Twitter platform data collection and preprocessing and implementing deep learning models to conduct text classification:

The first challenge we encountered is Twitter data collection. Since the Twitter API has low rate limits which restricts downloading huge amounts of tweets information, we explored python packages like Snsrape to gather data, while still confronting missing tweet issues. The authors also gained valuable experience in identifying potentially useful preprocessing steps specifically for tweet text, including removing substrings commonly used on Twitter, adding other common stop words from Twitter and using TweetTokenizer from nltk.tokenize package.

The authors also learned text feature extraction with the sklearn package and different architectures for NN models. We learned how to design embedding layers, hidden layers and etc. based on our own data set structure and gained valuable experience of building them with PyTorch on our own. We also explored hyperparameters tuning including number of hidden layers, size of embedding layers, pretrained embeddings, batch size and etc. to improve model performance, through which enhance our theoretical understanding.

From reading relevant research papers, we gained many great ideas of model selection and initiative ways of combining models for future work, adding CNN to BERT models for instance. Through conducting text classification, we mitigated data imbalance influence through upsampling and learned the idea of few-shot learning.

Future Work

Event Detection in News Articles

The problem of extracting events from news articles or other textual sources, sometimes referred to as Topic Detection (TD), part of Topic Detection and Tracking (TDT), or Event Detection (ED), has been much-attempted in the last few decades, as it has use cases in many domains. Panagiotou, Katakis, and Gunopulos[14] present a hierarchy of existing approaches, of

[14] N. Panagiotou, I. Katakis, and D. Gunopulos, "Detecting Events in Online Social Networks: Definitions, Trends and Challenges," in Solving Large Scale Learning Tasks. Challenges and Algorithms: Essays Dedicated to Katharina Morik on the Occasion of Her 60th Birthday, S. Michaelis, N. Piatkowski, and M. Stolpe, Eds. Cham: Springer International Publishing, 2016, pp. 42–84. doi: 10.1007/978-3-319-41706-6_2.

which clustering approaches make up a significant proportion. Other approaches include anomaly detection, which tries to detect outliers in texts based on models developed from more standard circumstances. Yet another approach is First Story Detection (FSD), which tries to identify novel events by the differences in texts from those of their neighboring documents. Finally, they consider approaches which focus on identifying particular topics, called topic specification.

There are many novel approaches being attempted to improve topic detection for news sources, such as the work of Xiao, Qian, and Qin[15] which uses a capsule semantic graph representation of news articles based on the relationship between semantic units of the document, calculates the similarity with the graph kernel, and performs incremental clustering of the news documents. Other work, by Rospocher et al.[16] develops event-centric knowledge graphs to represent long-term developments for events. However, replication of such approaches may be beyond the scope of this project, given our focus on NLP techniques and the other aspirations we have for our project.

A more relevant approach for the capabilities and time constraints may be the work by Jiang et al. (2021) which builds on historical TDT developments to develop a method called T-E-BERT that incorporates both temporal and content information into the representation of each news article. They demonstrate the application of their technique in two contexts: retrospective event detection for an existing corpus of texts, and online streaming for more real time analysis. For our purposes, we'll initially focus on their retrospective demonstration, though it would be an interesting and fruitful enhancement to our process to employ it in near real time.

T-E-BERT, their "time-text encoder", incorporates temporal information by converting the document publishing date into a time step, starting with the earliest document in the corpus, and transforming this into embeddings with sinusoidal positional encoding, repeated enough times to match the shape of the text embedding matrix which they create using BERT and an entity presence-absence embedding layer. The text and temporal matrices are fused using multi-head attention, and then this is trained on an event similarity task. This training uses a triplet loss function, in which an anchor document, a positive document from the same event as

[15] K. Xiao, Z. Qian, and B. Qin, "A graphical decomposition and similarity measurement approach for topic detection from online news," *Information Sciences*, vol. 570, pp. 262–277, Sep. 2021, doi: 10.1016/j.ins.2021.04.029.

[16] M. Rospocher et al., "Building event-centric knowledge graphs from news," Journal of Web Semantics, vol. 37–38, pp. 132–151, Mar. 2016, doi: 10.1016/j.websem.2015.12.004.

the anchor, and a negative document from a different event are utilized to compute a loss function which is composed of the cosine similarities between the anchor document and the positive and negative documents. They finally utilize this trained model to vectorize input news articles of interest and feed these vectors into the HDBSCAN clustering algorithm to identify clusters of events in the documents.

Additional recent research byMele and Crestani (2017) proposes an approach to detect events in heterogeneous news sources, which will also be relevant for our purposes as we incorporate the Twitter data utilized in the detection task to subsequently detect events while simultaneously detecting events from other news sources. Mele and Crestani note that event detection often is tailored to the type of text (for example, pulling events from short, informal tweets is optimized with different models and training than pulling events from full, published articles). They develop an approach called Event Detection and News Clustering (EDNC), which assumes a series of news from distinct sources (called ‘streams') as the input.

The first step of EDNC is to apply Latent Dirichlet Allocation (LDA) to generate topics and assign each text to the topic which is most represented in the text. They next use named-entity recognition to identify the frequency of certain event features (ie, named entities and phrases relevant to the event) within given time periods across the entire set of news texts. They keep the highest frequency event features in each temporal bucket and then compare the event features across buckets to delineate distinct events and the length over which they occurred. Any streams of news are then categorized first by filtering to only the relevant time range and by ranking the similarity of each text to the event features associated with a given event from training. They use cosine similarity as the similarity metric between news texts and event features.

Interpretation and feature importance for deep learning models

Though we applied feature importance functions for traditional supervised learning models, we have not implemented attributing the predictions of deep learning models to their input features. Understanding the input-output behavior of the deep neural networks could help us understand the strengths and weaknesses of the model and compensate for them. Attributions could also be used by developers in an exploratory sense. For instance, we could use a deep network to extract insights that could be then used in a rule based system. Sundararajan, M.,

Taly, A., & Yan, Q. (2017) use the axioms called integrated gradient to identify the feature importance. Unlike other proposed methods, integrated gradients do not need any instrumentation of the network, and can be computed easily using a few calls to the gradient operation, allowing even novice practitioners to easily apply the technique[17]

Combine CNN with BERT and LSTM models to improve performance

As our textbook states: Given the fact that CNN is in essence a feature-extracting architecture, CNN could be instead integrated into a larger network and trained to work in tandem.[18] Dong, J., He, F., Guo, Y., & Zhang, H. (2020) utilized Bert-CNN model to improve the accuracy of sentiment analysis models: BERT model was constructed firstly, and then a representation layer was input into the model to encode the review texts; after then, CNN semantic extraction layer was utilized to extract local features of the review text vectors, BERT semantic extraction layer to extract global features of the review text vectors and semantic connection layer to fuse features extracted by the two complementary models; in the end, a sentiment analysis of online commodity reviews was performed via the sentiment classification layer. The BERT-CNN model elevated F1 score by 14.4% and 17.4% respectively compared to BERT and CNN models.[19]

Combine other data sources to enlarge training set

As shown in previous section, when we used the Waseem, Z., & Hovy, D. (2016) dataset, which includes its own labels identifying racist and sexist tweets, to predict hateful comments using our LSTM model, we find that the model struggles to capture many of the tweets labeled as either racist or sexist. The reason could be the training set is relatively small and the corpus contained in the training set is limited. The other potentially useful datasets for future training and testing are as followings: Toxic Comment Classification Challenge, Jigsaw Unintended Bias in Toxicity Classification, the two Kaggle datasets published by Jigsaw; TweetEval Dataset

[17] Sundararajan, M., Taly, A., & Yan, Q. (2017, July). Axiomatic attribution for deep networks. In International conference on machine learning (pp. 3319-3328). PMLR.
[18] Goldberg, Y. (2017). Neural network methods for natural language processing. Synthesis lectures on human language technologies, 10(1), 1-309.
[19] Dong, J., He, F., Guo, Y., & Zhang, H. (2020). A commodity review sentiment analysis based on BERT-CNN model. In 2020 5th International Conference on Computer and Communication Systems (ICCCS) (pp. 143-147). IEEE.

published by Barbieri, F., Camacho-Collados, J., Neves, L., & Espinosa-Anke, L. (2020), which consists of seven heterogeneous Twitter-specific classification tasks (Emoji Prediction, Emotion Recognition, Hate Speech Detection, Irony Detection, Offensive Language Identification, Sentiment Analysis, Stance Detection).

Hyperparameters tuning

For deep learning models, we could implement more sets of hyperparameters to improve the model performance, including number of hidden layers, number of neurons in each hidden layer, pre-trained embeddings, learning rate, activation functions, etc.

Other methods to deal with data imbalance

Our model results show that using an upsampled dataset could significantly improve model performance. Though strategies used to mitigate imbalance in supervised learning, such as oversampling, can offer a stronger solution to the class imbalance problem, we would love to explore other methods to deal with data imbalance. Since it is time consuming and costly to get annotated data for Twitter hate speech, we could implement few-shot learning in future work, which aims to train models on a limited number of labeled samples given in a support set in order to generalize and classify unseen samples from a query set.[20]

[20] Ochal, M., Patacchiola, M., Storkey, A., Vazquez, J., & Wang, S. (2021). Few-shot learning with class imbalance. *arXiv preprint arXiv:2101.02523*.

Task Assignment

	Natalie Ayers	**Sophia Mlawer**	**Yangzhou Ou**
Proposal	Literature review on the preferred models and evaluation methods for event and topic detection from news sources.	Literature review on Google Trends and how its been used with hate speech, created Google Trends graphics	Literature review on Twitter hate speech classification, models and evaluation metrics selection
Data Collection & Exploration	Exploratory analysis on cleaned datasets; manual creation of vocabularies and feature sets	Used the Google Trends API to get the relevant temporal data and explored results	Tweet scraping pipeline, Tweet preprocessing pipeline, Other datasets exploration
Supervised Classifiers	Logistic regression using unigram/bigram count/binary feature sets	--	Bag-of-words and TF-IDF-based supervised classifier models pipeline (Logistic regression, multinomial Naive Bayes, Support Vector Classification, Linear Support Vector Classification, Gradient Boosting Classifier, XGBoost Classifier, Random Forest Classifier, and KNeighborsClassifier)
Deep Learning Models	distilBERT models	bi-LSTM models	CNN models
Prediction Testing	Cleaned new Tweet data; used saved bi-LSTM to predict hate speech with this data; qualitative analysis of different classifications; temporal comparison with Google Trends	Used original bi-LSTM code for this; created new Google Trends data for overlapping time period	--

Final Report	distilBERT model performance and explanation, data exploration prediction testing, future event detection sections; organized/partial writing of traditional ML models section	Bi-LSTM model performance and explanation, Google Trends section, the introduction, and organization of paper	Twitter theoretical review, Data Sources and Cleaning, TF-IDF, Bag-of-Words, and CNN models performance and explanation, Lessons Learned and Future Work

References

Watanabe, H., Bouazizi, M., & Ohtsuki, T. (2018). Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE access*, *6*, 13825-13835.

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 11, No. 1, pp. 512-515).

Liu, S., & Forss, T. (2015, November). New classification models for detecting Hate and Violence web content. In 2015 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K) (Vol. 1, pp. 487-495). IEEE.

Gambäck, B., & Sikdar, U. K. (2017, August). Using convolutional neural networks to classify hate-speech. In Proceedings of the first workshop on abusive language online (pp. 85-90).

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In Proceedings of the 26th international conference on World Wide Web companion (pp. 759-760).

Waseem, Z., & Hovy, D. (2016, June). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL student research workshop (pp. 88-93).

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, *10*(1), 1-309.

Ce, P., & Tie, B. (2020). An Analysis Method for Interpretability of CNN Text Classification Model. *Future Internet*, *12*(12), 228.

Wang, Z., & Qu, Z. (2017, October). Research on Web text classification algorithm based on improved CNN and SVM. In 2017 IEEE 17th International Conference on Communication Technology (ICCT) (pp. 1958-1961). IEEE.

Hajiabadi, H., Molla-Aliod, D., Monsefi, R., & Yazdi, H. S. (2020). Combination of loss functions for deep text classification. *International Journal of Machine Learning and Cybernetics*, *11*(4), 751-761.

N. Panagiotou, I. Katakis, and D. Gunopulos, "Detecting Events in Online Social Networks: Definitions, Trends and Challenges," in Solving Large Scale Learning Tasks. Challenges and Algorithms: Essays Dedicated to Katharina Morik on the Occasion of Her 60th Birthday, S. Michaelis, N. Piatkowski, and M. Stolpe, Eds. Cham: Springer International Publishing, 2016, pp. 42–84. doi: 10.1007/978-3-319-41706-6_2.

K. Xiao, Z. Qian, and B. Qin, "A graphical decomposition and similarity measurement approach for topic detection from online news," *Information Sciences*, vol. 570, pp. 262–277, Sep. 2021, doi: 10.1016/j.ins.2021.04.029.

M. Rospocher et al., "Building event-centric knowledge graphs from news," Journal of Web Semantics, vol. 37–38, pp. 132–151, Mar. 2016, doi: 10.1016/j.websem.2015.12.004.

Sundararajan, M., Taly, A., & Yan, Q. (2017, July). Axiomatic attribution for deep networks. In International conference on machine learning (pp. 3319-3328). PMLR.

Dong, J., He, F., Guo, Y., & Zhang, H. (2020). A commodity review sentiment analysis based on BERT-CNN model. In 2020 5th International Conference on Computer and Communication Systems (ICCCS) (pp. 143-147). IEEE.