Hate Speech Detection in Tweets

Rupal Jain, Sarah Song

Introduction

The project aims to tackle the prevalent issue of hate speech and abusive language in tweets, leveraging the popularity and widespread use of Twitter as a platform for expressing opinions, sharing thoughts, and engaging in conversations. As social media continues to play a significant role in shaping public discourse, it is crucial to address the rising tide of online hate speech and its negative impact on individuals and communities. With recent advancements in AI text generation, the concern further intensifies as these language models are trained on large volumes of online text, including hate speech, thereby potentially perpetuating biased and harmful content. This project focuses on developing mechanisms to detect hate speech in tweets and promote the generation of non-biased text. Different approaches that are based on the topics covered in the course were experimented for our specific task. We show a diagram that outlines our pipeline to detect hate speech in Fig 1, and the improved approach of this pipeline in Fig 2.

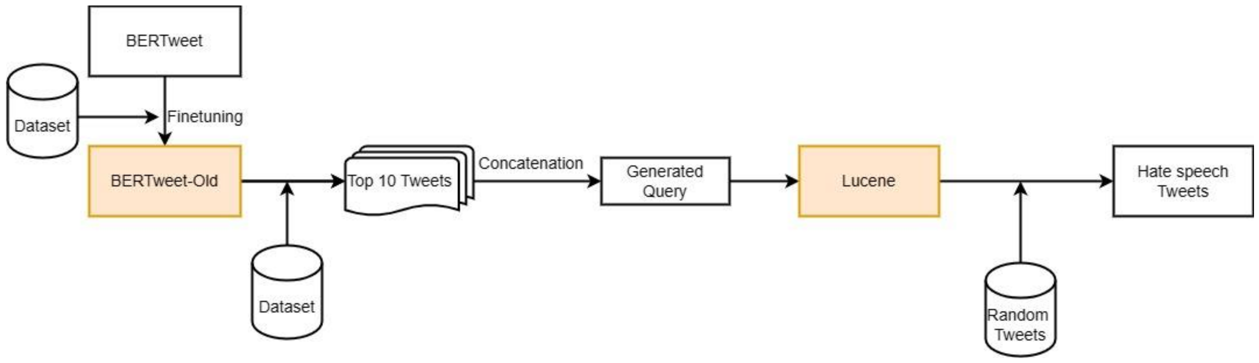


Fig 1: The diagram illustrates our pipeline for hate speech detection

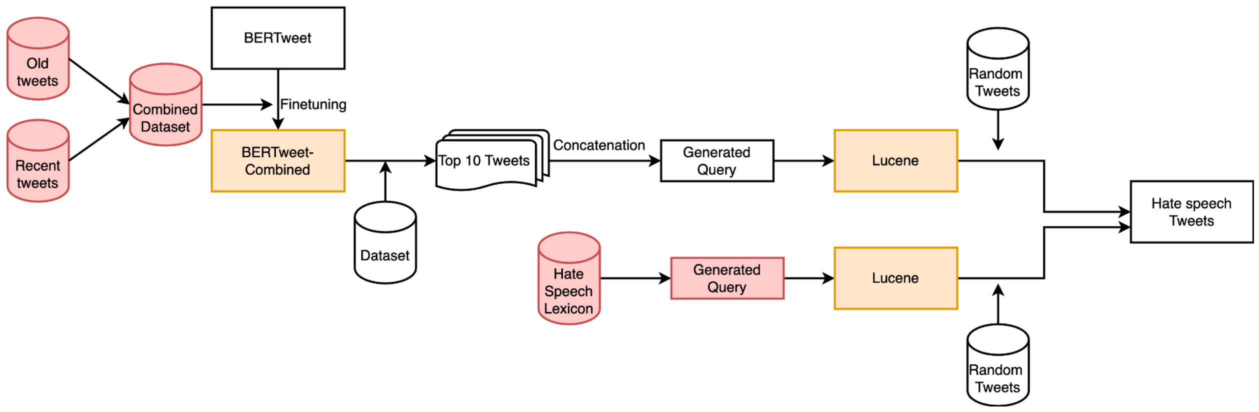


Fig 2: The diagram illustrates the enhanced iteration of the model, where the components highlighted in red have been incorporated to enhance the system's performance.

Motivation

The motivation behind this project is to address the pressing issue of hate speech within the context of tweets on the popular social media platform, Twitter. As a widely used medium for expressing opinions, sharing thoughts, and engaging in conversations on diverse topics, Twitter has unfortunately witnessed a surge in hate speech and abusive language. This escalating problem is further compounded by the recent advancements in AI text generation, which often rely on training data that includes such harmful content. It is crucial for language models to be equipped with the ability to generate non-biased text and contribute to a more inclusive online space. Thus, this project aims to explore various mechanisms for effectively detecting hate speech in tweets, ultimately fostering a safer and more respectful digital environment.

Description of the command line to run the code

Our code is available at
https://github.com/Rupaljain27/Hate_Speech_Detection_in_Tweets.

Neural Network in Jupyter Notebook

We used jupyter notebooks for the development of the neural network. In our github repository of the folder *neural network*, there are several jupyter notebooks that were used for the development of the classifiers.

- Improved_Neural_Network_Text_Retrieval(CSC583)_Project.ipynb : contains the fine tuning process of the BERTweet on our datasets to develop a hate speech classifier.
- Combined_Neural_Network.ipynb : contains the finetuning on the combined dataset to develop a hate speech classifier.
- HateSpeech_Classifier.ipynb: contains the code that uses the fine tuned model to classify the test dataset directly
- Hate_Speech_Lexicon_Build.ipynb : contains the code that generates the lexicon query string for the improvement method of the Lucene model approach.

Lucene in Java

To run the Java code from the command line, follow the steps below:

1. Firstly, copy the below repository to your local directory by executing:

   ```
   git clone
   https://github.com/Rupaljain27/Hate_Speech_Detection_in_Tweets.git
   ```

2. Open the terminal or command prompt and navigate to the folder where the repository has been cloned using the cd command. For example, if the repository is cloned in the "project" folder, you can navigate to it using:

   ```
   cd project
   ```

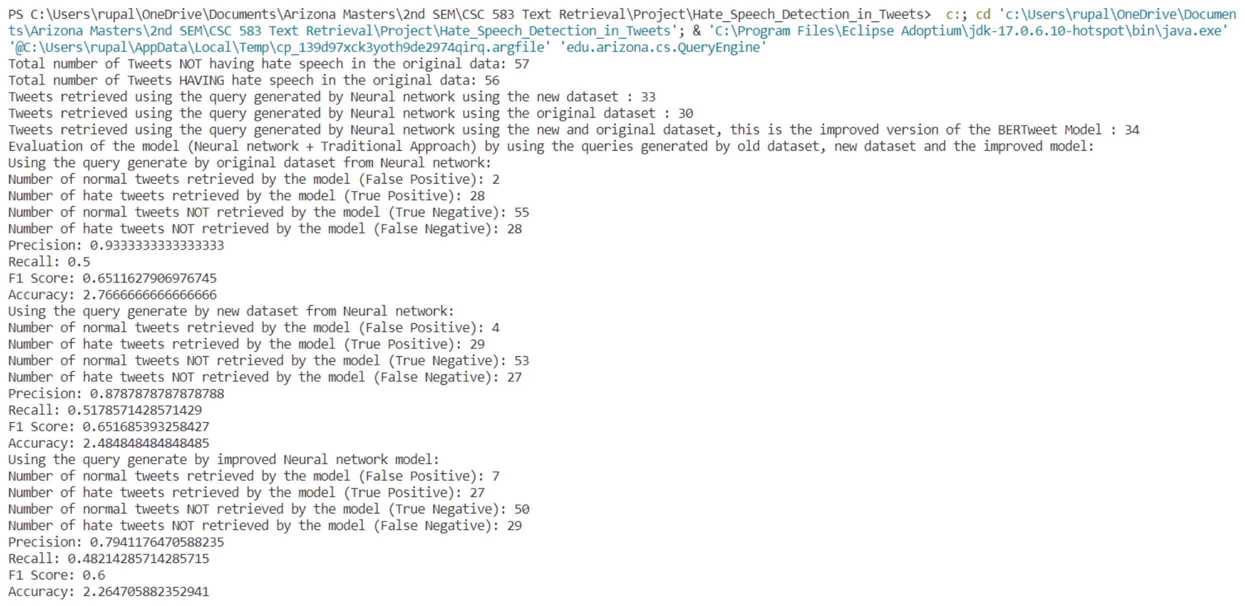
3. Once you are in the correct folder, execute the following command to compile the Java code:

   ```
   javac src/main/java/edu/arizona/cs/QueryEngine.java
   ```

   This command compiles the Java source file QueryEngine.java located in the specified path.
4. After the compilation is successful, you can run the Java program using the java command followed by the fully qualified class name. For example, if the main class is QueryEngine located in the edu.arizona.cs package, you can run it using:

   ```
   java edu.arizona.cs.QueryEngine
   ```

A snap of the code execution is attached below, the output shows the precision, recall, f1-score and accuracy of the complete model using the 3 types of queries given by the neural network: the query generated by original dataset, new dataset and combination of original and new dataset.



```
PS C:\Users\rupal\OneDrive\Documents\Arizona Masters\2nd SEM\CSC 583 Text Retrieval\Project\Hate_Speech_Detection_in_Tweets>  c:; cd 'c:\Users\rupal\OneDrive\Documen
ts\Arizona Masters\2nd SEM\CSC 583 Text Retrieval\Project\Hate_Speech_Detection_in_Tweets'; & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.6.10-hotspot\bin\java.exe'
'@C:\Users\rupal\AppData\Local\Temp\cp_139d97xck3yoth9de2974qirq.argfile' 'edu.arizona.cs.QueryEngine'
Total number of Tweets NOT having hate speech in the original data: 57
Total number of Tweets HAVING hate speech in the original data: 56
Tweets retrieved using the query generated by Neural network using the new dataset : 33
Tweets retrieved using the query generated by Neural network using the original dataset : 30
Tweets retrieved using the query generated by Neural network using the new and original dataset, this is the improved version of the BERTweet Model : 34
Evaluation of the model (Neural network + Traditional Approach) by using the queries generated by old dataset, new dataset and the improved model:
Using the query generate by original dataset from Neural network:
Number of normal tweets retrieved by the model (False Positive): 2
Number of hate tweets retrieved by the model (True Positive): 28
Number of normal tweets NOT retrieved by the model (True Negative): 55
Number of hate tweets NOT retrieved by the model (False Negative): 28
Precision: 0.9333333333333333
Recall: 0.5
F1 Score: 0.6511627906976745
Accuracy: 2.7666666666666666
Using the query generate by new dataset from Neural network:
Number of normal tweets retrieved by the model (False Positive): 4
Number of hate tweets retrieved by the model (True Positive): 29
Number of normal tweets NOT retrieved by the model (True Negative): 53
Number of hate tweets NOT retrieved by the model (False Negative): 27
Precision: 0.8787878787878788
Recall: 0.5178571428571429
F1 Score: 0.651685393258427
Accuracy: 2.484848484848485
Using the query generate by improved Neural network model:
Number of normal tweets retrieved by the model (False Positive): 7
Number of hate tweets retrieved by the model (True Positive): 27
Number of normal tweets NOT retrieved by the model (True Negative): 50
Number of hate tweets NOT retrieved by the model (False Negative): 29
Precision: 0.7941176470588235
Recall: 0.48214285714285715
F1 Score: 0.6
Accuracy: 2.264705882352941
```

Description of the code

Neural Network

1. **Indexing and Retrieval**

We use the transformers for the neural network implementation of the project. Our project problem deals with data – tweets, that have a specific characteristic, hence we selected the pre-trained model BERTweet (Nguyen, 2020), which is a language model trained on English tweets fetched from 2012-2019. As we are using a transformer network, we do not have a separate process for indexing the documents. We conduct fine tuning on different datasets for our hate speech detection task. The dataset that we used for this task is as follows, and is contained in the Dataset folder of our github repository.

- Dataset A: A dataset that contains 31962 tweets with hate speech that was generated in 2018. These tweets were fetched in a similar time range of the tweets that BERTweet was initially trained on.

- Dataset B: A dataset that contains 19934 tweets with hate speech that was generated in 2020. This dataset was selected to have the model be finetuned on more recent tweets.

- Dataset C: A dataset that is used to generate hate speech query in the next step of the pipeline. This data contains 24783 tweets that were generated in a similar time range of Dataset A.

Due to the class imbalance problem of the above datasets, we conduct cost-sensitive training during the fine-tuning. As the distribution of the datasets are biased towards non-hate speech labeled tweets, we utilize the class weights to give more weight on penalizing the misclassification of the hate speech labeled tweets. This logic is implemented in the *CustomTrainer* of our code in Improved_Neural_Network_Text_Retrieval(CSC583)_Project.ipynb. The finetuning and evaluation is contained in the same notebook as well.

2. **Measuring performance**

After splitting the dataset into a ratio of train:dev:test = 6:2:2, we conduct hyperparameter tuning of number of epochs, weight decay, batch size, and learning rate on the dev set. Using the best performing hyperparameters, we fine tune the models on the dataset.

2.1. BERTweet fine-tuned on dataset A (BERTweet-Old)

2.1.1 Evaluation results on test dataset of A

	Precision	Recall	F1-score
Non-hatespeech	0.98	0.99	0.98
Hate-speech	**0.66**	**0.65**	**0.65**

2.1.2 Evaluation results on dataset C

	Precision	Recall	F1-score
Non-hatespeech	0.81	0.95	0.87
Hate-speech	**0.32**	**0.10**	**0.15**

2.2. BERTweet fine-tuned on dataset B (BERTweet-New)

2.2.1. Evaluation results on test dataset of B

	Precision	Recall	F1-score
Non-hatespeech	0.96	0.90	0.93
Hate-speech	**0.50**	**0.74**	**0.60**

2.2.2. Evaluation results on dataset C

	Precision	Recall	F1-score
Non-hatespeech	0.79	0.72	0.76
Hate-speech	**0.19**	**0.25**	**0.22**

When evaluating and comparing the fine tuned models on the dataset C, we see that BERTweet-Old has a comparatively higher precision score, and BERTweet-New has a higher recall score.

3. Scoring function

We use the standard metrics of Precision, Recall, and F1-score as we are dealing with an unranked problem for this step of the project. We use these metrics to evaluate the model on the test split of the dataset that it is finetuned on, and also compare the fine-tuned model's performance by applying it on dataset C.

4. Improving Retrieval

After observing that the BERTweet-Old has a strength of having high precision of predicting hate speech tweets, and BERTweet-New has a strength of having high recall of predicting hate speech tweets, we decide to merge these two datasets together in the expectation that this will produce a model that has the advantages of BERTweet-Old and BERTweet-New. The implementation process of this is contained in Combined_Neural_Network.ipynb.

- Model fine tuned on a combined dataset, that consists of one copy of dataset A and one copy of dataset B (BERTweet-Combined)

4.1.1. Evaluation results on test dataset of A

	Precision	Recall	F1-score
Non-hate tweet	0.97	0.97	0.97
Hate tweet	**0.64**	**0.64**	**0.64**

4.1.2. Evaluation results on test dataset of B

	Precision	Recall	F1-score
Non-hate tweet	0.96	0.92	0.94
Hate tweet	**0.55**	**0.71**	**0.62**

4.1.3. Evaluation results on dataset C

	Precision	Recall	F1-score
Non-hate tweet	0.80	0.78	0.79
Hate tweet	**0.22**	**0.24**	**0.23**

We also attempted to give more emphasis on the dataset B, which contains tweets that are fetched recently. Due to time constraints we were not able to tune the proportion of the dataset B and dataset C in the combined dataset. However, when testing on a combined dataset which contains two copies of dataset B, the fine tuned model did not show improved performance. This might be due to the fact that dataset A and dataset C contain similar date ranges of tweets, but dataset B contains more recent tweets so emphasizing the dataset B degrades the performance. More excessive tuning and experiments will need to be conducted in future.

- Model fine tuned on a combined dataset, that consists of one copy of dataset A and two copies of dataset B

4.1.4. Evaluation results on test dataset of A

	Precision	Recall	F1-score
Non-hate Tweets	0.97	0.99	0.98
Hate Tweets	**0.74**	**0.55**	**0.63**

4.1.5. Evaluation results on test dataset of B

	Precision	Recall	F1-score
Non-hate Tweets	0.95	0.95	0.96
Hate Tweets	0.69	0.63	0.66

4.1.6. Evaluation results on dataset C

	Precision	Recall	F1-score
Non-hate Tweets	0.80	0.87	0.83
Hate Tweets	**0.21**	**0.14**	**0.17**

We observe that the model that is finetuned on the combined dataset maintains the performance when applying it to the test splits of dataset A and B. Also, by combining the two datasets we are able to see that the model has a more balanced precision and recall score when compared to BERTweet-Old and BERTweet-New.

Lucene Model

1. Indexing and Retrieval

In our project, we leveraged the powerful Lucene library to handle the indexing and retrieval of tweets. Our workflow began with the collection of hate and normal random tweets through the Twitter API, which were then stored in a CSV file. These tweets were subsequently indexed and stored in a dedicated directory using Lucene. During the indexing process, we applied techniques such as tokenization and normalization to ensure the tweet text was properly processed, resulting in an optimized index structure that enables efficient searching.

For tweet retrieval, we adopted both traditional text retrieval methods and an improved approach. In this method, we employed boolean search queries to determine the presence of hate speech in a tweet. These queries were derived from a neural network model, and we explored three variations for the finetuning process: using the original dataset, using a new dataset for model improvement, and a combination of the original and new datasets. Applying these fine tuned models on dataset C, we retrieve the top 10 tweets that the model has the highest probability(confidence) of having hate speech by applying softmax. Then, we concatenate the content of those top 10 tweets to generate a query string that will be used in Lucene. Additionally, we incorporated an enhanced retrieval approach by leveraging lexicons. We compiled a set of words known to be associated with hate speech and constructed a query using these words. The set of words were determined by intersecting two sets of hate speech lexicon that was available online. One set was a restricted version of the Hatebase lexicon (Davidson et al., 2017), and another set was a set of hate speech lexicons that were generated by various pre-existing resources (Bassignana et al., 2018). We use static word embeddings, Word2Vec, to expand the intersection of these two sets. For each hate speech keyword in the intersection, we get the top 5 word embeddings that are similar to it using the Glove-Twitter word embeddings available through the gensim library, and add them to the lexicon. Then, we generate a query that is a concatenation of this expanded hate speech lexicon. This process is implemented in

Hate_Speech_Lexicon_Build.ipynb. As result, the retrieval process involved matching the queries against the indexed tweets, retrieving relevant tweets, and assessing the presence of hate speech.

Overall, our project employed Lucene's indexing capabilities to efficiently organize the tweet collection, and we employed both traditional and improved retrieval methods to identify tweets containing hate speech. This comprehensive approach allowed us to navigate through the tweet data effectively and gain insights into the prevalence of hate speech within the collection.

Libraries:

1. **Lucene**: Lucene, a Java-based open-source search library, offers extensive indexing and searching capabilities. It enables developers to create efficient indexes for storing and retrieving textual data, ensuring speedy and precise search results. Lucene supports advanced features like tokenization, stemming, and scoring, facilitating techniques such as fuzzy matching, phrase queries, and relevance ranking. By utilizing Lucene, we harnessed its robust indexing mechanism to effectively organize and locate specific tweets within our large collection.
2. **Twitter4j**: Twitter4J is a Java library that provides an easy-to-use interface for accessing the Twitter API. It simplifies the process of integrating Twitter functionality into Java applications by abstracting away the complexity of the API endpoints and authentication mechanisms. Using Twitter4J, we can easily perform various tasks such as tweeting, searching for tweets, retrieving user information, and interacting with timelines.

Functions description:

The main file of the model is QueryEngine.java. There are 5 files which includes the functions to generate random tweets using Twitter4j, build index, search for the tweets using queries and evaluation metrics.

a) **RandomTweetsFetcher**.java

   This file contains the implementation for fetching random tweets from the Twitter API **'Twitter4j'** and performing some text normalization. Here's a breakdown of the code:

   1. It has a 4 variables for storing Twitter API keys and access tokens.
   2. The method getTwitterInstance() is used to create and return a Twitter instance with the configured API keys and access tokens.

3. The method normalizeText(String inputText) takes an input text and performs text normalization. It uses a NormalizeCharMap to define a mapping for the apostrophe, creates a CharFilter to remove the apostrophe, creates an instance of the StandardAnalyzer, tokenizes the input text, and builds a normalized text by iterating through the tokens.
4. The method **getRandomTweetsFromAPI**(int numTweets) is responsible for retrieving random tweets from the Twitter API. It creates a Twitter instance and has 2 ways to retrieve random tweets: retrieve the 20 most recent tweets from the home timeline and fetch random tweets from the Twitter API using a lexicon query generated by us. This query has a bunch of words identified as hate words.
5. After receiving the tweets, the code proceeds to normalize them and save them into a CSV file.

b) **IndexBuilder**.java

The purpose of this file is to construct an index. It begins by establishing an index writer and configuring it with a standard analyzer. Subsequently, an index writer configuration is generated. Moreover, a CSV reader is instantiated to retrieve the data from the input CSV file. The code proceeds to iterate through each CSV record using a CSV parser. Within this loop, a fresh Lucene document is generated, and both the text and Tweet ID fields are appended to the document. Finally, the document is added to the index using the index writer.

c) **HateSpeechDetector**.java

This file contains functions for detecting hate speech by utilizing a hate speech lexicon query along with neural network generated query and searching an index of tweets.

HateSpeechDetector(String queryString, String name): This function is responsible for detecting hate speech. It starts by creating an index and initializing file names for output. It opens an index reader and sets up buffered writers for writing the results. The function searches the index based on the provided query string, retrieves the matching tweets, and stores the results in a list. It also performs an improved method by searching the index using a hate speech lexicon. The results from both methods are combined and written in an output file.

d) **Evaluation**.java

The Evaluation class in the Output.java file contains functions for evaluating the performance of a hate speech detection model.

1) **Evaluation**(ResultFilePath_New,ResultFilePath_Combined, ResultFilePath_Original): This method performs the evaluation by comparing the retrieved tweets from different datasets with the original input data. It takes three file paths as input:

ResultFilePath_New for the new dataset, ResultFilePath_Combined for the combined dataset, and ResultFilePath_Original for the original dataset. It reads the ground truth hate labels (which is the total number of tweets having hate speech, similarly for normal tweets) from an input CSV file generated before and retrieves the tweets from the result files. It then calculates and prints various evaluation metrics using measuringPRA function..

2) **mearusingPRA**(groundTruthHate0, groundTruthHate1, retrievedTweets): This method calculates the evaluation metrics Precision, Recall, F1 Score, and Accuracy. It calculates the number of true positives, false positives, false negatives, and true negatives based on the retrieved tweets and original data for each of the 3 resultsets.
3) **getHateValues**(Map<String, Float> retrievedTweet, String filePath, int value): This method retrieves the number of tweets having hate values as 0 or 1 using the original data based on the retrieved tweets.

2. Measuring performance

Measuring the performance of the project involves employing various evaluation techniques to assess its effectiveness. Several key steps are taken to quantify the system's performance and ensure its accuracy and reliability.

a. **Evaluation Metrics**: The project utilizes well-established evaluation metrics such as Precision, Recall, and F1-score to measure the system's performance. These metrics provide a comprehensive understanding of the system's ability to correctly classify and retrieve relevant information.
b. **Test Dataset**: A carefully curated test dataset is used to evaluate the system's performance. The test dataset allows for objective measurement and comparison of different approaches or models.

Dataset: Tweets extracted using Twitter4j which include approx 50% as normal and 50% as hate tweets.
Total number of Tweets NOT having hate speech in the test dataset: 57
Total number of Tweets HAVING hate speech in the test dataset: 56

Applying traditional approaches on random test dataset extracted using Twitter API with generated query from Neural network

A. Applying query generated from BERTweet-Old
Number of normal tweets retrieved by the model (False Positive): 5
Number of hate tweets retrieved by the model (True Positive): 22
Number of normal tweets NOT retrieved by the model (True Negative): 52
Number of hate tweets NOT retrieved by the model (False Negative): 34

	Precision	Recall	F1-score
Hate Tweets	0.81	0.39	0.53

B. Applying query generated from BERTweet-New
Number of normal tweets retrieved by the model (False Positive): 6
Number of hate tweets retrieved by the model (True Positive): 34
Number of normal tweets NOT retrieved by the model (True Negative): 51
Number of hate tweets NOT retrieved by the model (False Negative): 22

	Precision	Recall	F1-score
Hate Tweets	0.85	0.60	**0.70**

C. Applying query generated from BERTweet-Combined
Number of normal tweets retrieved by the model (False Positive): 14
Number of hate tweets retrieved by the model (True Positive): 26
Number of normal tweets NOT retrieved by the model (True Negative): 43
Number of hate tweets NOT retrieved by the model (False Negative): 30

	Precision	Recall	F1-score
Hate Tweets	0.65	0.46	0.54

3. Scoring function

Within the traditional model, the performance of the system is evaluated using standard metrics such as Precision, Recall, and F1-score to tackle an unranked problem in the project. The evaluation takes place on a dataset obtained through the Twitter4j API. The metrics are computed for two systems: the first system involves searching for tweets using a neural network query, while the second system combines the neural network query with a lexicon query to search for tweets. By comparing the performance of these two systems, it is determined whether the model has undergone improvements or not.

4. Improved Retrieval

The improved approach involves incorporating lexicon words associated with hate speech to enhance the performance of the system. By constructing queries based on this set of lexicon words, we aim to improve the retrieval of relevant tweets containing hate speech. Additionally, we utilize a combination of results obtained from both the lexicon and neural network generated queries, exploiting the strengths of each method. This combined approach is applied to the different query types generated by the neural network model. Through the integration of lexicon-based retrieval and neural network-generated queries, our goal is to achieve a significant enhancement in overall performance, ultimately enabling more effective identification and assessment of hate speech in tweets.

The performance evaluation of the combined model entails the utilization of diverse evaluation techniques to gauge its effectiveness and ascertain its reliability. To quantify the system's performance, several crucial steps are undertaken.

a. **Evaluation Metrics**: Well-established evaluation metrics such as Precision, Recall, and F1-score are employed to assess the system's performance comprehensively. These metrics offer valuable insights into the system's capability to accurately classify and retrieve pertinent information.
b. **Test Dataset**: A meticulously curated test dataset is employed to evaluate the system's performance. This test dataset enables an objective measurement and comparison of various approaches or models. By utilizing this dataset, the system's performance can be objectively analyzed and benchmarked against alternative methods or models.

Dataset: Tweets extracted using Twitter4j which include approx 50% as normal and 50% as hate tweets.
Total number of Tweets NOT having hate speech in the test dataset: 57
Total number of Tweets HAVING hate speech in the test dataset: 56

- *Applying traditional approaches on random test dataset extracted using Twitter API with neural network and lexicon query.*

A. Combining the results from applying the query generated from BERTweet-Old and applying the lexicon generated query on the test dataset
Number of normal tweets retrieved by the model (False Positive): 2
Number of hate tweets retrieved by the model (True Positive): 28
Number of normal tweets NOT retrieved by the model (True Negative): 55
Number of hate tweets NOT retrieved by the model (False Negative): 28

	Precision	Recall	F1-score
Hate tweets	0.93	0.50	**0.65**

B. Combining the results from applying the query generated from BERTweet-New and applying the lexicon generated query on the test dataset
Number of normal tweets retrieved by the model (False Positive): 4
Number of hate tweets retrieved by the model (True Positive): 29
Number of normal tweets NOT retrieved by the model (True Negative): 53
Number of hate tweets NOT retrieved by the model (False Negative): 27

	Precision	Recall	F1-score
Hate Tweets	0.87	0.51	0.65

C. Combining the results from applying the query generated from BERTweet-Combined and applying the lexicon generated query on the test dataset
Number of normal tweets retrieved by the model (False Positive): 7
Number of hate tweets retrieved by the model (True Positive): 27
Number of normal tweets NOT retrieved by the model (True Negative): 50
Number of hate tweets NOT retrieved by the model (False Negative): 29

	Precision	Recall	F1-score
Hate Tweets	0.79	0.48	**0.60**

Through our observations, we noted that the model demonstrates improvements when utilizing the original dataset or a combination of the new and original datasets. However, it is worth mentioning that when employing the new dataset alone, the model's performance does not exhibit significant enhancement.

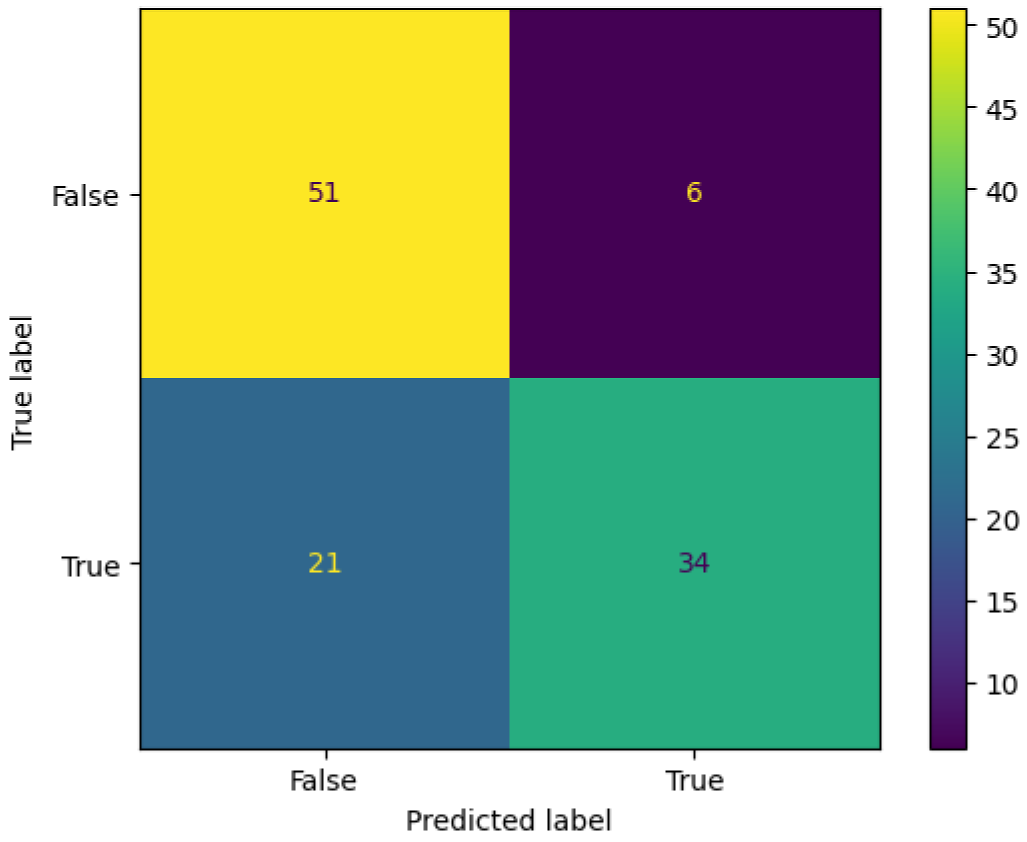
Classification

We also decided to compare applying the fine tuned model directly to the test dataset, without going through the process of applying Lucene. Among the different fine-tuned models we created from the previous step, we selected the best performing model : BERTweet-Combined, which was the one that was trained on the combined dataset of A and B. We apply this model on our curated test dataset to see how this model can work as a hate speech classifier directly, without the step of applying Lucene. The implementation of this step is in HateSpeech_Classifier.ipynb. The performance of this approach is shown in the below table. From the results, we see that directly applying the fine tuned model itself is not effective compared to our above approach.

	Precision	Recall	F1-score
Non-hate tweet	0.51	0.93	0.66
Hate tweet	0.50	0.07	0.13

Error Analysis

We selected the best-performing system, which was using the query generated from the model fine-tuned on dataset B. The confusion matrix of the model when applied to the test dataset is as below:



How many questions were answered correctly/incorrectly?

From the above confusion matrix, we see that the system was able to classify a total of 34 (TP) +51 (TN) = 85 tweets correctly. The system had 6 tweets that it classified as hate label when it was not (FP), and 21 tweets that it labeled as non-hate speech when it was actually a tweet containing hate speech (FN), so in total 37 tweets were misclassified.

What problems do you observe for the questions answered incorrectly? Try to group the errors into a few classes and discuss them.

We observed the false positives and false negatives, and saw that the errors fall into the below cases:

- Overlapping words with the query
  For the false positives, we focus on the tweets where the system assigned a high score but the tweet is labeled as non-hate speech. We observe that these tweets contain specific words that were repeatedly contained in the generated query. For example, in the below tweet

  we have sacred obligation care men women who fought protect our freedom how maga republicans congress have chosen repay them

  This tweet contains the word men and women, and the generated query had repeated occurrences of the words "men" and "women". Hence even though this tweet does not contain hate speech and is describing a situation or an event, the system gave it a high score and misclassified it.

- Implicit hate speech
  For the false negatives, we see that the tweet does not have an explicit individual or group of people they are expressing hate. For example, in the below tweet

  rt mrdavemacleod theyve never done anything theyre fucking idiotsits going week total royal mania coronation

  The author of this tweet is expressing a form of violence to the British royal family, but this is not mentioned explicitly in the tweet by referring to them as "they". The model is fine tuned on a dataset that contains hate speech tweets that contain explicit targets of the hate speech - such as keywords of "woman", "man", "white", and "black". As the given tweet does not have a target of the hate speech clearly mentioned, the system was not able to classify this as a hate speech tweet.

Conclusion

In this project, we were able to experiment with different concepts that were covered in our course. We developed a pipeline that uses the transformers network and the Lucene library. From the results, we see that using the query generated from BERTweet-New shows the best performance. This is likely to be due to the fact that our curated test dataset contains hate speech of recent tweets, and BERTweet-New is fine-tuned on tweets that are relatively recent than other datasets. This implies that the dialogue of hate speech is continuously changing in the web and accounting to that fact is important for the system to be able to retrieve hate speech tweets.
We also observe that using our pipeline has a better performance compared to directly applying the fine tuned model as a classifier of hate speech. This shows that combining different approaches can improve the system performance.

There are some limitations in our approach, in that we were not able to test excessively on hyperparameter tuning due to resource and time constraints.

References

- Bassignana, E., Basile, V., & Patti, V. (2018). Hurtlex: A multilingual lexicon of words to hurt. *Proceedings of the Fifth Italian Conference on Computational Linguistics CLiC-It 2018*, 51–56. https://doi.org/10.4000/books.aaccademia.3085
- Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1), 512–515. https://doi.org/10.1609/icwsm.v11i1.14955
- Nguyen, D. Q., Vu, T., & Tuan Nguyen, A. (2020). Bertweet: A pre-trained language model for English tweets. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. https://doi.org/10.18653/v1/2020.emnlp-demos.2