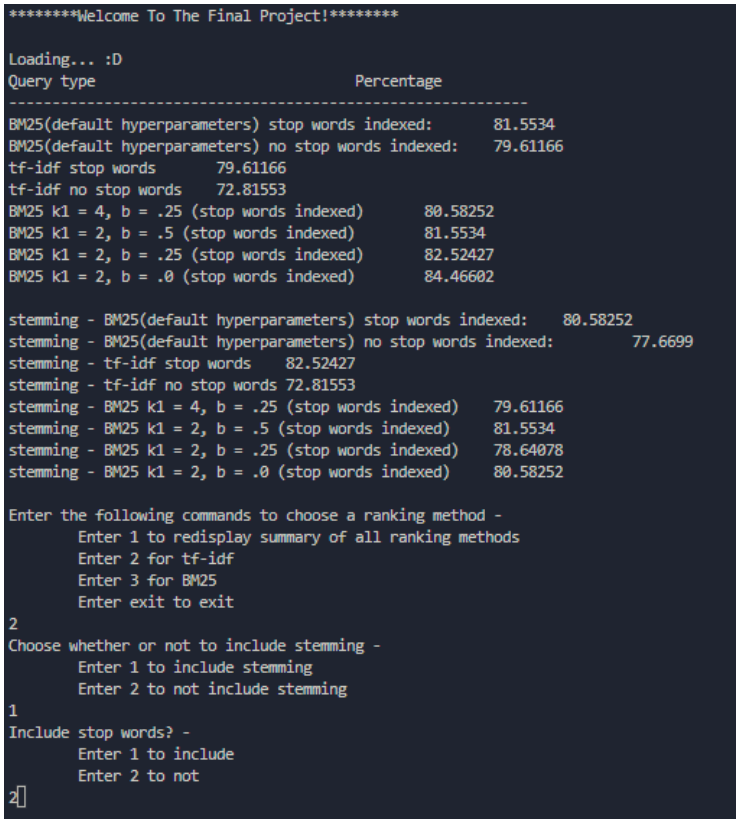
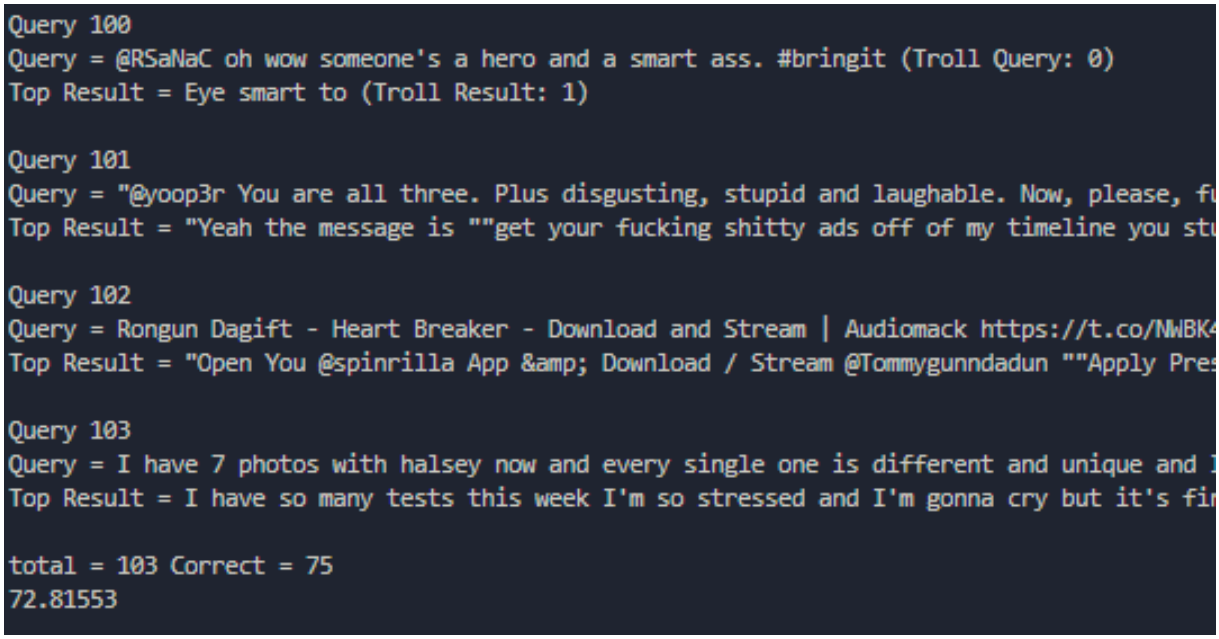
Eric Simonson, Zachary Lopez, JayGee Galvez

Motivation:
Social media has played an increasingly important role in today's civil/political society. Because of this malicious actors can exploit these platforms through the use of "trolling" operations to help sow civil discord and confusion. For example, Russian "trolling" operations for influencing online political discourse have been prevalent for the past decade (Matt Burgess). As such, the ability to detect these sorts of trolls would be very useful in helping to weed out these bad actors on social media sites. Therefore, the purpose of this project is to construct ways of detecting these trolls.

Description of command line:

- For the Java/Lucene portion of the code, the main class is QueryEngine.java.
- The CSV files for the query and index are located in the resources folder.
- This program does not require any command line arguments. Once the program is running, it will display the percentages for each scoring system to include BM25, tf-idf, indexing stop words, stemming, and BM25 hyperparameters. The terminal will ask the user to input different commands to customize which specific search settings they want. When it is done asking for user input it displays the raw results of the query using the specified scoring system showing each query with the top result that matches. A 0 indicates a non-troll tweet and 1 indicates a troll.
- Example of program running
- 

- User specifies the parameters, 2 for tf-idf, 1 to include stemming, 2 to not include stop words.
- 

Description of code:

- For our Java program, we used Lucene.
- We compiled a csv containing 40,000 tweets, 20,000 of which are verified Russian troll tweets and 20,000 from accounts that weren't associated with Russian groups. The Russian troll data set was gathered by 538 who worked with 2 professors from Clemson University Darren Linvill and Patrick Warren (FiveThirtyEight). The accounts who authored the tweets in the data set were connected to the Russian troll factory the Internet Research Agency. The other 20,000 non-troll tweets were from Data Society (Twitter User Data - Dataset by Data-Society).
- Each line contains two columns - the first column contains the tweet content and the second column contains 0 or 1 (0 indicates a non-troll tweet and 1 indicates a troll).
- We removed 100 tweets from this dataset to use as queries (otherwise, the system would match 100% of the time).
- The main Java class for the program is QueryEngine.java.
- Lucene builds the index from the remaining ~40,000 tweets. A QueryEngine object is created for each scoring/tokenization strategy (BM25, tf.idf, indexing stop words, stemming, and varying BM25 hyperparameters). For each strategy, the tweets contained in the query csv are compared to the index. For each tweet query, Lucene finds the top result and uses the "troll value" (second column value) as the categorization of the tweet.

The troll values of the query and the top result are compared. A count is kept of correctly categorized tweets and the overall percentage correct is calculated.

Questions:

1) For the index we used Lucene's StandardAnalyzer for non-stemming analysis and EnglishAnalyzer for stemming. We found that non-stemming indexes yielded better results. When stemming was used, we found that a higher number of troll tweets were correctly identified, but more non-troll tweets were incorrectly marked as troll tweets (false positives).
   In regard to stop words, our results were better when stop words were included in the index. This may be attributed to non-native English speakers using more stop words than native speakers (Jane Im).
   For the queries, we included urls and hashtags, as the StandardAnalyzer and EnglishAnalyzer both indexed these. Also, including urls was another datapoint for comparison between troll and non-troll tweets. Russian trolls include urls in their tweets more often than non-trolls. We did not include emoji or other unknown characters as there was not a straight-forward way to use them with Lucene.
   Otherwise for the query, we used the entire content of the tweets, and did not create a subset. The tweets themselves were a short length and creating a subset would have removed pertinent information (as we saw with the relevance of stop words).
   One problem with tweet data is that when extracting the tweets to be used for the queries instead of indexing, some of the tweets were re-tweets and not identified as duplicates because the content of the re-tweet differs slightly from the original tweet. These differences are usually in the embedded url and sometimes the re-tweet includes an additional or different hashtag.

2) We used P@1 for categorization of the tweets. Using this performance metric, our best result was 84.5% for BM25 with hyperparameters k1 = 2 and b = 0.
   The top result was used to categorize the tweet as troll or non-troll.
   We also considered taking the average value of the top ten tweets to categorize the queried tweet, but we did not think that would be relevant because our goal was categorization (binary) rather than multiple results.

3) We compared BM25 with tf.idf. We found that tf.idf performed worse than BM25 (79.61% vs 84.5%). This is due to tf.idf giving more weight to rare terms. In tweets, the terms we are looking for may not be all that rare and many of the tweets use similar language/terms. Our queries are also using the entire text of the tweet, so many of the query terms will not be rare, but we still want to match them to the documents in the index. BM25 performs better than tf.idf because it takes the document's length into

account along with term frequency in the document. Some terms, such as stop words, are not rare, but we still wanted to match them. In tf.idf, since stop words are common in the documents and the collection as a whole, they are assigned less weight than other terms. With BM25, we were able to control the term frequency value with the k1 hyperparameter.

4) With BM25 (k1 = 2, b = 0) with no stemming, our system correctly categorized tweets 84.5% of the time. Some of this may be attributed to the fact that many of the troll tweets contain political content, thus overfitting could be a problem for our system. Another factor may be that since many of the trolls are non-native English speakers, they use stop words more frequently than native speakers (Jane Im). Since the Lucene analyzers index urls, another contributing factor could be the fact that trolls are more likely to include urls in their tweets.
5) The alternative approach uses the Python library Keras to use machine learning to achieve the same task of determining if a given tweet is a bot troll account or legitimate. All the code to this can be found in the python directory. To run the code run the following bash commands:

```
pip install -r python/requirements.txt
python3 python/process_data.py --remove-stopwords --lemmatize
python3 python/main.py
```

   a) The first command ensures all Python libraries are correctly installed.
   b) The second command runs the process_data.py script which preprocesses the data to remove stop words and lemmatize all words.
   c) The third command executes the main program and once finished displays the results summary.
   d) The provided values from an example run are provided (Removed stop words and lemmatized):

```
Loss:  0.4862779974937439
Accuracy:  0.6322386860847473
F1 Score:  0.7793806791305542
Precision:  0.8053498864173889
Recall:  0.7664137482643127
```

   e) From this we can see that our results have a relatively low accuracy, however it yields a high F score which means this method provides answers with high confidence.

Works Cited

FiveThirtyEight. “Russian Troll Tweets.” *Kaggle*, 1 Aug. 2018, https://www.kaggle.com/datasets/fivethirtyeight/russian-troll-tweets.

Ollie. “Why We're Sharing 3 Million Russian Troll Tweets.” *FiveThirtyEight*, FiveThirtyEight, 31 July 2018, https://fivethirtyeight.com/features/why-were-sharing-3-million-russian-troll-tweets/.

“Twitter User Data - Dataset by Data-Society.” *Data.world*, 3 Dec. 2016, https://data.world/data-society/twitter-user-data.

Im, Jane, et al. ‘Still out There: Modeling and Identifying Russian Troll Accounts on Twitter’. *12th ACM Conference on Web Science*, Association for Computing Machinery, 2020, pp. 1–10, https://doi.org10.1145/3394231.3397889. WebSci ’20.

Burgess, Matt. “We Finally Know the Full Extent of Russia's Twitter Trolling Campaign.” *WIRED UK*, WIRED UK, 17 Oct. 2018, https://www.wired.co.uk/article/twitter-troll-data-russia-ira-iran.