

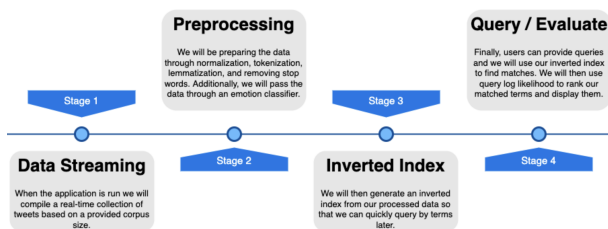
Real-Time Twitter Opinion Retrieval and Querying

Caleb Johnson
Information Retrieval
Salt Lake City, UT
calebdeejohnson@gmail.com

Introduction

Twitter is a social media networking application with millions of concurrent users. It stands as a meeting place and cross-section for opinions, confessions, and musings, with an average of 500 million new tweets sent out each day. The application can give us a rough snapshot of the hivemind, and I am curious in exploring these results.

In this project, I am creating a retrieval system that will first build a novel, real-time corpus of tweets reflecting the current state of opinions on the site. Then I will process the data, classify each tweet by multiple metrics, and then build an inverted index for querying. Users will be able to query key terms and retrieve a ranked list of matching tweets to their query, as well as apply filtering on the results based upon emotion, sentiment, length, and more. This general pipeline is represented below and will be discussed throughout this paper.



Motivation

Emotion and opinion analysis is a vital part of marketing and analytics for firms and individuals alike. Being able to see what people are saying and feeling about keywords in real-time can be both educational and critical for fast-paced industries.

This could answer questions such as "What emotions are people feeling about Apple's ongoing press release" by building the corpus as the presentation is in progress, and then querying

by key word's "Apple, MacBook, Event" . This would allow us to observe the emotions Twitter users are feeling in the moment in regards to this. It could also be used for something as simple as to see how people are feeling about dogs at a certain point of the day.

Novelty and Related Work

Analysis of Twitter data is a frequent area of research, as it is one of the most data-rich environments available to people. However, I couldn't find any literature relating to creating a novel dataset for querying to gain insight into current discussion topics.

Emotion classification of Twitter tweets is also a common area of research, and there has been some work done in real-time analysis. In Rik Van de Walle and Sofie Van Hoecke's work they write a chapter entitled *A Learning Based Approach for Real-Time Emotion Classification of Tweets* which I referenced for building my emotion classification model in PyTorch [1].

Data Collection

This project will employ novel, real-time datasets constructed each time the application is run. Utilizing the Twitter API with developer credentials, we will stream a set number of tweets to create this dataset when the application is run. The user will provide a corpus size, with a default size of 1,000, and our streamer will run until this quantity is achieved. My current Twitter developer credentials allow me to stream 200,000 tweets in a single stream, so given my limitations that is the maximum size of the corpus.

After we have streamed our desired number of tweets, we will pass each tweet through a trained emotion neural network that will provide us with a multi-class label for each tweet. We will then go through the steps of processing our data: first removing stop words, then normalizing, then intelligently lemmatizing our text for each tweet. Finally, we will tokenize each tweet so that we have a list that we can iterate through. Once we have finished our pre-processing steps, we will create an inverted index, where we have a dictionary, with each observed word as a key and then an index to each tweet that contains that keyed word as a part of a list of tweets. This step will allow us to quickly build a list of tweets that match one of the keywords in our queries, which will speed up our query times when we run the model later.

Experimental Design

After formulation of the dataset, users are able to query the corpus to see current opinions on different topics. Users will be able to provide keywords that they would like to query by, as well as further filtering terms for their results such as emotion type and tweet length. We will be able to use our inverted index to pull all the tweets that matched one or more of the keywords in our query. We will then create a relevance ranking that will be used to order the matching tweets in the display for the user, and to do this we will compute a query likelihood value for each tweet that matched a term in our query. Equation 1 shows the basic formula for the query likelihood function, which will use a basic maximum likelihood estimation to give each document a score.

$$\sum_{p(t|\theta_d)} \log(p(t|\theta_d)) \quad (1)$$






The corpus created at the application's run-time will have very variable sizes, and will definitely not be representative of all the concepts the user may be querying on. To account for potential zero probability concerns while computing the query likelihood, we will employ Dirichlet Smoothing (see equation 2). This smoothing function will find both the p_{MLE} for the term in the retrieved tweet as well as the corpus as a whole. It will weigh these two likelihoods by the size of the retrieved tweet and also provide a slight bias term λ to further prevent zero probability problems if the term still doesn't appear in our corpus. After tuning, I have assigned the hyperparameter μ a value of 5, which is about the standard deviation for words in a tweet.

$$p(t|\theta_d) = \frac{|d|}{|d| + \mu} p_{MLE}(t|d) + \frac{\mu}{|d| + \mu} p_{MLE}(t|C) \quad (2)$$

Results

After the model has computed the query log likelihood score for each tweet that contained one of the keywords, they are ordered by their score. For simplicity, we display the top ten matching tweets to the user. An example for how this may look can be seen below in *Example Query: Biden Kennedy Center Honors*.

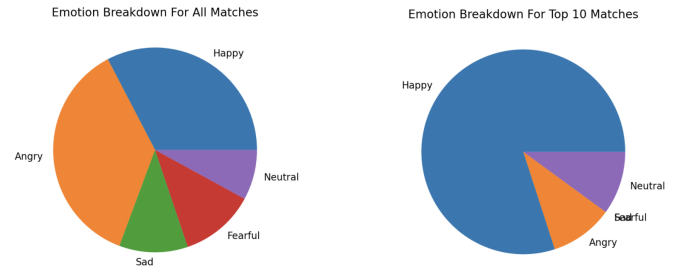
Example Query: "Biden Kennedy Center Honors"

	"Return to tradition: Biden celebrates Bette Midler, Joni Mitchell at Kennedy Center Honors" <small>Mon Dec 06 07:40:05 +0000 2021</small>	Emotion: Happy	QLL: -11.09
	"RT @thedailybeast: Biden gets standing ovation at Kennedy Center Honors after four years of Trump boycott" <small>Mon Dec 06 08:05:49 +0000 2021</small>	Emotion: Happy	QLL: -11.56
	"RT @AwardShowNews: #JillBiden & #JoeBiden at the 44th Kennedy Center Honors #KennedyCenterHonors https://t.co/1I9EUax3Ra " <small>Mon Dec 06 08:12:05 +0000 2021</small>	Emotion: Neutral	QLL: -18.21
	"Scarlett Johansson scintillates in low-cut sequined gown with Colin Jost at Kennedy Center Honors" <small>Mon Dec 06 07:50:15 +0000 2021</small>	Emotion: Happy	QLL: -18.74
	"Biden Sends A Message By Restoring Kennedy Center Honorees White House Reception https://t.co/uY5OY1KSVl " <small>Mon Dec 06 08:10:01 +0000 2021</small>	Emotion: Happy	QLL: -21.25

Total Matched Tweets: 95
Median QLL: -38.91






The Kennedy Center Honors was an event that was going on while I built my dataset on December 5, 2021. In the event, Joe Biden made an unexpected appearance that made headlines, as a president hadn't attended the event in

four years. Looking at the top five results included below, we see that our model was able to grab events that very closely reflected what we were looking for. In total, 95 tweets included one of the key words "Biden", "Kennedy", "Center", and "Honors", though our model provided a ranking that seemed to closely reflect the most relevant examples.



The pie charts above show the emotion profiles of our matched tweets, the first being all 95 matched tweets, while the second shows only our ten most relevant tweets from our query log likelihood ranking. Four out of the top five tweets by this ranking were classified as happy, and eight of the top ten overall. When we look at the entire set of matched tweets, however, our profile gets much wider. Among all tweets, angry overtakes happy as the predominant emotion. This helps us to see that Twitter users are feeling relatively happy about Biden's appearance at the Kennedy Center Honors event, however, the wider community has much more varied opinions about him in general.

Example #2: "Omicron Variant Coronavirus Covid"

	"RT @timesofindia: Omicron variant live updates: 94 students of Karnataka school test Covid positive amid Omicron alarm..." <small>Mon Dec 06 07:16:42 +0000 2021</small>	Emotion: Fearful	QLL: -20.33
	"RT @timesofindia: Omicron variant live updates: 94 students of Karnataka school test Covid positive amid Omicron alarm..." <small>Mon Dec 06 07:16:32 +0000 2021</small>	Emotion: Fearful	QLL: -20.33
	"RT @SkyNews: BREAKING: South African President, Cyril Ramaphosa says the Omicron COVID variant appears to be dominating new infections in m..." <small>Mon Dec 06 08:00:33 +0000 2021</small>	Emotion: Sad	QLL: -21.61
	"RT @CBSNews: JUST IN: Omicron COVID variant was in Europe before South African scientists detected and flagged it to the world" <small>Mon Dec 06 07:50:15 +0000 2021</small>	Emotion: Neutral	QLL: -21.79
	"RT @EssexPR: So, we have Covid pill treatments becoming available within weeks, the Omicron variant is looking like it's very mild and has..." <small>Mon Dec 06 08:10:01 +0000 2021</small>	Emotion: Happy	QLL: -21.96

Total Matched Tweets: 299
Median QLL: -35.95

A second example can be seen above in *Omicron Variant Coronavirus Covid*. In this example, we were querying for something a bit more broad that wasn't attached to a single ongoing current event. One decision choice is highlighted in this example as we see the top two tweets in our ranking are retweets of the same tweet. I ultimately decided to include retweets, though the decision could well be made to not include them in our dataset formulation.

Evaluation

As we are generating a novel dataset on each run of our application, we have no ground truth to compare our model's ordering against. While our orderings may have seemed reasonable to us, we wanted to have a way to evaluate their quality. To do this, I used normalized discounted cumulative gain.

Discounted cumulative gain is a metric to evaluate an orderings performance given relevance labels. In regards to our model, our query log likelihood function provided relevance outputs that we used to order our tweets and return them. To evaluate our model, however, we needed actual labeled data to check.

I ran 10 different queries without seeing the query-log likelihood ordering. The model used our inverted index to return every tweet that included one of keywords from the query. I then hand labeled them based off how I perceived the relevance of the tweet given the query using a 5-level system. Finally, I used these labels to compute our models discounted cumulative gain, the ideal discounted cumulative gain, and lastly report our normalized discounted cumulative gain for each query. In regards to calculating DCG as well, I only used the ten most relevant tweets (for our model, the ten most relevant tweets from the query log likelihood calculation).

Query #1: "Oklahoma Sooners Football Brent Venables"

Relevance: 5	Relevance: 2	Relevance: 1
RT @espn: The Sooners got their guy !! Clemson DC Brent Venables is Oklahoma's new head football coach.	@sidlowe It'd be mainly about Spanish football culture(my friends and I are Culers), and journalism in sports.	@business_brent @Buy_Brent Brent free data collection app for third party always making money at the resident expense!
Query Log Likelihood Ordering	Ideal Ordering	
Relevance: 5	Relevance: 5	DCG: 14.819
Relevance: 5	Relevance: 5	iDCG: 14.889
Relevance: 5	Relevance: 5	nDCG: 0.995
...	...	
Relevance: 1	Relevance: 1	

Above we have two query examples, both of which were trending topics on Twitter that afternoon. The first query was in regards to the University of Oklahoma's football team hiring their new head coach, Brent Venables. Our first example, "RT @espn The Sooners got their guy !! Clemson DC Brent Venables is Oklahoma's new head football coach", is an example of a tweet that received a perfect relevance score of 5. Likewise, the next example was a tweet that was only marginally relevant, while the third and final example held no relevance at all. Our model's ordering performed very well, and had a discounted cumulative gain of 14.819, while the ideal ordering would have been 14.889. This means we an nDCG score of 0.995.

Query #2: "Steph Curry Warriors Basketball NBA"

Relevance: 5	Relevance: 3	Relevance: 1
Warriors Steph Curry Just Did Something We May Never See Again...	Timberwolves shoot for more success at home vs. Hawks #Timberwolf #NBA #AllEyesNorth #PowerOfThePack	RT @magulangsofanji: #AboutAnji Did you know that Anji plays volleyball, basketball and badminton? Yes, she is a sporty chic.

Another example query was "Steph Stephen Curry Warriors Basketball NBA", which performed similarly to our first query. Overall, it had an nDCG of 0.936.

Table 1. nDCGs of the 10 Queries

Query	Matches	nDCG
Oklahoma Sooners Football Venables	59	0.995
Steph Curry Warriors NBA Basketball	39	0.936
Angela Merkel Retire	28	0.856
Scarlett Johansson Kennedy Center	44	0.903
Vladimir Putin Russia Ukraine	81	0.885
Bob Dole Dies	39	1.000
Omicron Variant Spike Europe	421	0.812
Utah Jazz Rudy Gobert	29	0.936
Christmas Season Shopping	711	0.915
Snow Storm Utah Colorado Wyoming	397	0.894
Average		0.913

The above table shows the ten different queries that we calculated the discounted cumulative gain for our model, the ideal DCG, and the normalized DCG for our model's performance. When we average the nDCG that we found for all ten queries, we get an average nDCG value of 0.913.

Conclusion and Future Works

This has been a fascinating project to work through over the course of the semester. I have been pleased with the results I have been able to obtain so far, but there are quite a few other parts I want to explore.

Currently, the query log likelihood function is completely oblivious to the proximity of the key terms within tweets. It weights everything equally, which provided interesting results, however I would like to improve the model by better weighting by context proximity in the model. Consider the query "Kennedy Center Honors", where two matched tweets are "John Kennedy was at the center of the politics" and "Kennedy center was a fantastic experience I must say" are both 9 words in length, with 1 case of two different key terms. They would have the same query log likelihood score, but I would like to give greater preference to tweets where the key terms are closer together.

One factor that I haven't been able to properly account for in my evaluation was inherently ambiguous queries. For instance, all of my evaluation queries were very specific and this made it easy to determine whether the matched tweets were relevant or not. A simple query, such as, "java" would be more difficult to evaluate, however, because of so many potential meanings. To get a more robust level of query types to evaluate on as well we can download a dataset with has relevance labels included.

The model is entirely run from within the command line currently, however I would like to build out a better interface to interact with the model in. We output our ranking to a csv that we have to look through, but I would like to be able to display them in a GUI.

Appendix - Run Code

The repository can be accessed at <https://github.com/Calebdee/IR-RealTimeTwitterQuery> and cloned to your local machine. The main files

Before you will be able to run the application you will need to sign up for a free Twitter developer account and generate your own access tokens. These will then need to be provided in the `twitter_credentials.py` file before you will be able to run the program. Once you have generated your own API tokens, you will need to first build the dataset by running `./build_model.sh`, and you can further provide the size of the model by providing it as the following command line argument. For instance, a corpus size of 50,000 would be `./build_model.sh 50000`.

Once the model has finished generating the dataset, you can run `./query.sh "QUERY TERMS HERE"` to search for a given query. The full ranking of matched tweets can be viewed in the `out.csv` file that is generated each time we run our query script.

Works Cited

[1] Janssens O., de Walle R.V., Hoecke S.V. (2015) A Learning Based Approach for Real-Time Emotion Classification of Tweets. In: Kazienko P., Chawla N. (eds) Applications of Social Media and Social Network Analysis. Lecture Notes in Social Networks. Springer, Cham. https://doi.org/10.1007/978-3-319-19003-7_7