An Algorithmic Approach to Trending Tweet Prediction and Recommendation

Bharath Srivatsan and Kelly Zhou
Andrea LaPaugh

Abstract

Social media has become an integral component of our daily lives, both as a source of news and information and as a method of connecting with others. Concurrently with the rise of social media activity and growth in the variety of available online platforms, Twitter has emerged as a prominent means of connecting with friends, following celebrities and events, and engaging in topical conversations with users from around the world. One of the many features Twitter offers is a list of trending topics and hashtags which represent popular subjects of conversation at any given time. For users who wish to expand their following, discovering soon-to-be trending topics is one of the most important ways they can spread their message to others. In this paper, we propose early detection algorithms for predicting topics and hashtags that will soon trend, and develop a method of recommending pre-trending topics based on individual user preferences.

1. Introduction

With internet usage rising every year, social networking has become the most popular online activity in the United States [7]. Of the many social-networking and real-time micro-blogging sites that have emerged in recent years, Twitter is the most popular. Users of all ages and backgrounds use Twitter to connect with others, follow and share news, and stay up to date with daily trends. With strict text length limits of 140 characters per tweet, Twitter uses its social-networking to enable rapid topic diffusion, and the common practice of retweeting further draws attention to select ideas. As a result, Twitter has been uniquely able to centralize and motivate massive discussions on so-called 'trends.' Through its constantly evolving trending list of words, phrases, and hashtags (words with the # sign affixed to them), updated based on constant real-time tweet activity, users can identify relevant and

prominent themes at any given time and follow conversations accordingly. Following trends in turn allows users to both learn about new topics that would be absent from a 'local feed'-based social media application (like Facebook) and also discover new perspectives from individuals who are from completely different social circles. In any case, following trending lists on Twitter enables users to stay informed, relevant, and connected with the broader online community.

Twitter is also a key means of personal and financial development. Celebrities and corporations alike command massive followings on the social media application, and users who wish to break into this world often aim to do the same. Since Twitter serves as a casual way of connecting with an incredibly diverse audience, building a large following is vital. [12]

1.1. Goal

The goal of this project is twofold: first, we hope to predict topics and hashtags that will become trending before they actually trend, and second, we aim to recommend pre-trending topics to users based on their individual tweet history and preferences. To accomplish this goal, we implement and evaluate five different methods to predict emerging trends from a corpus of over a million real-time tweets. After collecting our list of predicted pre-trending topics, we then recommend a selection of them to users based on information derived from previous Twitter activity.

1.2. Motivation

Twitter users often work to gain and maximize visibility online, whether through building their follower base or by maximizing impressions on tweets. Users who join trending conversations early on generally enjoy increased social media recognition and visibility, as the trending topics often lead wider Twitter communities to their relevant tweets. As such, the ability to detect trending topics early would enable users to engage in conversations before they come to worldwide prominence, and in turn would put users at the forefront of these movements. We thus find trend prediction analysis to be a promising means of enabling users to maximize their Twitter visibility.

Moreover, Twitter often serves to introduce users to new topics and information sources. One of the main reasons Twitter identifies trends is to provide a curated list of potentially interesting

items to users. Not all trending information aligns with the interests of the different members of the diverse Twitter community, though, and as such, we identify the utility of a recommendation engine. Beyond predicting upcoming trends, recommending potentially trending topics to users would introduce them to ideas that are closely linked to their interests. A recommendation engine can also be used as a tool for users to leave footprints on growing discussions that they have vested interests in, thereby magnifying their impact.

With thorough prediction and recommendation analysis, we thus identify a promising opportunity to optimize the user experience on Twitter through increased visibility and personal interest alignment.

2. Related Work

With the rise in social-networking, the study and analysis of social media activity has received much attention as a means of understanding underlying sociological patterns. Given its trending classification and social tagging features, Twitter has become a common model and platform for such prediction analysis, and many studies have been conducted on trending topic and hashtag prediction.

Some studies have attempted to predict the popularity of new hashtags and topics with various machine learning and classification models, including Naive Bayes, k-nearest neighbors, decision trees, support vector machines, and logistic regression. In one report published by researchers at Stanford University and Trinity University, the authors propose using Euclidean distance to assess the similarity between tagging systems to predict hashtags [15]. Here, we find tweets represented as points in a high dimensional space and a network constructed by the latent space model [15]. This algorithm utilizes a distance function to address a major challenge in hashtag prediction, the rapidly changing Twitter landscape with new hashtags developing constantly, which often makes classification models involving clustering difficult. In a separate paper published by researchers at the National Laboratory of Pattern Recognition in Beijing, we find a method of Twitter topic trend prediction based on Moving Average Convergence-Divergence (MACD), which is commonly

used to predict trends in stock prices [13]. In this method, trends are predicted based on the average occurrences of key words over different time frames [13]. One of the most notable methods of trend prediction was developed by MIT professor Devavrat Shah, who proposed a nonparametric approach that applies machine learning algorithms to compare real-time tweets with previously trending tweets using distance calculations weighted according to a decaying exponential model [5].

Given a wide variety of methods used to analyze and predict trends within the machine learning world, we incorporate different aspects of previous prediction algorithms to devise our own series of methods to implement and evaluate.

In addition to previous work in prediction analysis, substantial work with respect to recommendation systems also exists. In a report published by researchers at Singapore Management University, we find a survey of recommender systems used in Twitter [14]. Here, we find traditional recommender systems based on collaborative filtering and content-based recommendation enhanced to cater to specific Twitter recommendations. The paper also introduces different methods of hashtag recommendation using a TF-IDF scheme, a Bayesian model, and a high-dimensional Euclidean space model. For the purposes of this paper, we take these different methodologies into consideration to create a recommendation engine feasible for the scope of this project.

3. Prediction Analysis

The first component of our project aimed to develop a series of algorithms that could predict soon-to-be-trending topics and hashtags. We built five parallel systems to do this, each approaching the problem from a different direction. All of our methods used Twitter's Streaming API service to collect a corpus of tweets and ultimately returned a ranked list of 50 topics they 'predicted' would soon be trending. We then evaluated these rankings against actual Twitter trends scraped over the following few hours.

For each of the methods we used, we've described our approach, implementation, results, and evaluations. Unfortunately, since the results themselves are typically in the form of massive data files that include huge lists of words/hashtags and their attributes, we aren't able to post the full lists

here or online. We've included small snippets of the top results for each of the methods below.

3.1. Method 1: Hashtag Ranking

3.1.1. Approach Our first method was the simplest – it ranked the most commonly used 'hashtags' seen within each corpus of tweets. Trending topics can be in the form of words, phrases, and hashtags, but users typically pick hashtags with care since they serve as representations of the message they're trying to convey. As such, hashtags often reach the trending page (whenever their underlying message is picked up by others and made viral). [10]

Rank by: Presence-in-tweets

3.1.2. Implementation See commented code at Hashtag Frequency Algorithm.

In order to collect the tweets needed to determine hashtag frequencies, we needed to interface with the Twitter Streaming API (we used Tweepy, an easy-to-use wrapper, to do this)[4]. Of the two ways to stream tweets in real time – a sample method to pull a random assortment of tweets from the firehose and a query method to continuously pull all fresh tweets that satisfy a set of parameters – we went with the latter in order to limit our dataset to English language tweets. Since we still wanted to collect as many tweets as possible, we set about constructing a query that would capture almost all English language tweets from the firehose[9]. By collecting terms from the 500 most commonly used words on Twitter (including stop words like 'a', 'the', and 'like'), we were able to query almost all English tweets [11]. While we may be missing some tweets from the firehose sample, we don't see any reason to believe that the small proportion of tweets we miss will coincide with any particular trends that we can't identify elsewhere (i.e., that our broad sample isn't generally representative of Twitter). See code here: Data Stream Code [6]

After collecting the tweets, we built a dictionary for tags to track the number of occurrences of each hashtag. Because the datasets were so large (for this method, we used 200,000 tweets), our initial implementations hit a variety of memory errors and had to be rewritten and optimized. Ultimately, this meant that we had to abandon a previous approach that used simpler lambda

mapping within data frames in favor of modularized for loops and individual lists/dictionaries. See old version here: Old Data Analysis Algorithm

3.1.3. Results Over the course of one hour, we collected 200,000 tweets as our test sample. Running our algorithm on those tweets yielded the following results:

Rank	Prediction	Relevance
1	ALDUBinITALYDay2	Irrelevant
2	RecountVP	Highly Relevant
3	BBM4VP	Highly Relevant
4	SoccerAid	Irrelevant
5	LIKEIWOULD	Highly Relevant
6	FCBayern	Slightly Relevant
7	RT	Irrelevant
8	Gameinsight	Slightly Relevant
9	MGWV	Irrelevant
10	androidgames	Irrelevant

3.1.4. Conclusion A little surprisingly, this crude initial method was quite successful at discovering trends! This is likely because a relatively small proportion of tweets actually use hashtags, so there might be a lower bar for hashtags to make it onto the trending list. The relative paucity of hashtag-rich tweets also detracted from this method, though. The vast majority of tweets in our sample did not include hashtags, and so even though we had 200,000 tweets, the diversity and volume of hashtags we saw was far lower. A more robust prediction engine would require a larger corpus of tweets to capture greater diversity and make predictions with greater accuracy. With our bandwidth and storage limitations, however, it was incredibly difficult to collect much larger collections of tweets that were not too spread apart in time. For this algorithm to work, it would have to pick up on hashtags that were active over a small span of time; the fact that the Twitter streaming API limited its transmissions due to our slow connection meant that it took us a long time to collect this selection (and so we missed short term trends).

That said, there are a variety of open questions regarding potential optimizations to this method. For one, we could have weighted the hashtags based on how many other hashtags a user posted in the same tweet, or whether the hashtag was positioned ahead of others in the tweet's text. It might also have been fruitful to discount hashtags that were tweeted by a limited number of users – it

seems like trending topics would be those talked about by a wide variety of users.

3.2. Method 2: Word Ranking

3.2.1. Approach This method ranks the most commonly seen words in the corpus of 200,000 tweets we collected. Like the hashtag method, this algorithm is fundamentally based on counting the instances of topics in the text of a tweet. However, moving from hashtags to words created a new set of challenges regarding stop words and inappropriate words (but also brought in far more data, allowing for more reasoned predictions).

Rank by: Presence-in-tweets

3.2.2. Implementation See commented code at Term Frequency Algorithm

Utilizing the same techniques as in the hashtag algorithm to stream tweets and construct a dictionary, we were able to modify the hashtag code into text code without much difficulty. By iterating through all of the words within each given tweet, we were able to capture a wide variety of possibilities for our dictionary.

That said, there were three new challenges we faced. First, stop words dominated the rankings. Since words like 'a' are so central to English, almost all tweets include them, and so they naturally rise to the top of a count-based ranking. By filtering out these words before they entered our dictionary, we solved this problem [3]. Similarly, there were a variety of words that had to be eliminated based on their content. A large number of tweets included pornographic references and terms; knowing that Twitter would never let such inappropriate topics reach the trending list, we manually collected the top 15 or so most common words of this kind from our sample dataset and scratched them as well. Similarly, we eliminated Twitter-related words (like followers).[1] See eliminated words here: Scratch Words. Finally, in order to ensure uniformity, we erased all punctuation and converted every word into its lower-case form.

[1] Interestingly enough, the most common word used in our sample set was RT (an abbreviation for retweet). A whopping 92,000 tweets from our 200,000 tweet sample were retweets!

3.2.3. Results Over the course of one hour, we collected 200,000 tweets as our test sample. Running our algorithm on those tweets yielded the following results:

Rank	Prediction	Relevance
1	love	Irrelevant
2	people	Irrelevant
3	amp	Irrelevant
4	mom	Slightly Relevant
5	happy	Irrelevant
6	jeremyelowe	Irrelevant
7	workfromhome	Irrelevant
8	5hbbmas	Irrelevant
9	aldubbonvoyage	Irrelevant
10	school	Irrelevant

3.2.4. Conclusion The performance of this algorithm was not impressive. Because this algorithm simply returned the words associated with the highest volume of tweets, information on changes in their usage or analysis of the kinds of users who tweeted them was absent. Furthermore, unlike the hashtag algorithm that was limited to a smaller set of terms consciously chosen by users, the word-based method indiscriminately returned terms from all parts of user tweets.

The largest stumbling block that exists for this method is likely differentiating trending words from common words. The list of top words often includes banal terms like 'love' that aren't stop words but are still not relevant to Twitter trends. It seems that any automated method to differentiate between words that might make Twitter's trending list and words that are just fillers in English will require more information than just frequencies - manually excluding words doesn't seem to be a viable option.

3.3. Method 3: Authority Weighting

3.3.1. Approach This method weights terms based on the kinds of users who tweet them. It takes into account a given term's presence in hashtags and frequency in tweets, but also the number of verified users who have used it and the maximum number of followers of any user who tweeted about it. Weighting tweets by the authoritativeness of their users will hopefully create a more intelligent prediction algorithm; given that famous individuals or users with a wide variety of followers are more likely than others to be able to catapult tweets into the trending list, weighting

topics that have been 'picked up' by celebrities seems intuitively more likely to result in an accurate ranking. To do this, we calculated the relevant factors, applied scaling factors to them, normalized them, and combined them in a final, weighted algorithm:

$$\text{Rank By: } (\alpha * \text{freq}_{x,mod} + \beta * \text{follow}_{x,mod} + \gamma * \text{ver}_{x,mod})\delta$$

where:

$$\text{freq}_{x,mod} = \frac{log(log(\text{freq}_x)) - \mu_{log(log(\text{freq}))}}{\sigma_{log(log(\text{freq}))}}$$

$$\text{follow}_{x,mod} = \frac{log(\text{follow}_x) - \mu_{log(\text{follow})}}{\sigma_{log(\text{follow})}}$$

$$\text{ver}_{x,mod} = \frac{log(\text{ver}_x) - \mu_{log(\text{ver})}}{\sigma_{log(\text{ver})}}$$

$$\delta = \begin{cases} 4 \text{ if term present in hashtag} \\ 1 \text{ otherwise} \end{cases}$$

See other metrics returned by the Twitter JSON dump here: Sample JSON Output

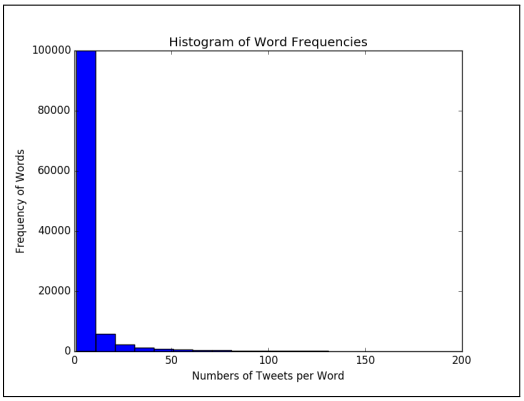
3.3.2. Implementation See commented code at Authority-Weighting Algorithm

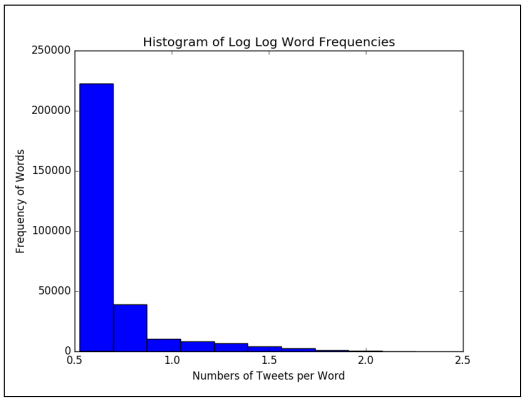
We first computed hashtag and text frequencies based on the algorithms developed in the previous two methods. This time, though, as we collected each word, we also kept track of user attributes. We maintained separate dictionaries to note the largest number of followers of any user who posted a given word and to note the number of verified users who used each term.

Since each of these metrics were based on different scales, we had to normalize the scores before we could combine them. However, the histograms of the given values for term frequency, max follower frequency, and verified user numbers were far from normally distributed.

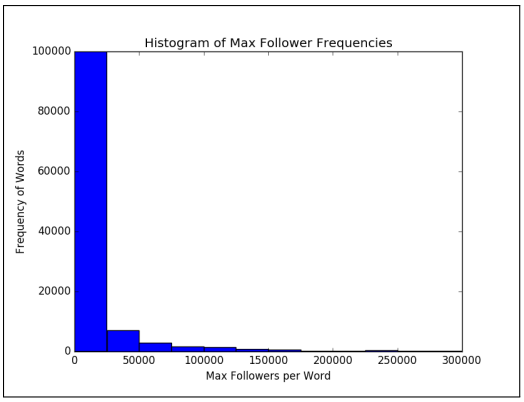
Each diagram was highly skewed-right, with the highest volume of values clustered around 0 and a long tail stretching to numbers many orders of magnitude higher. In order to correct for this, we took the logarithms of the values. For the first metric (term frequencies), taking the logarithm once wasn't enough to normalize the data, so we ended up taking another logarithm to reduce the impact

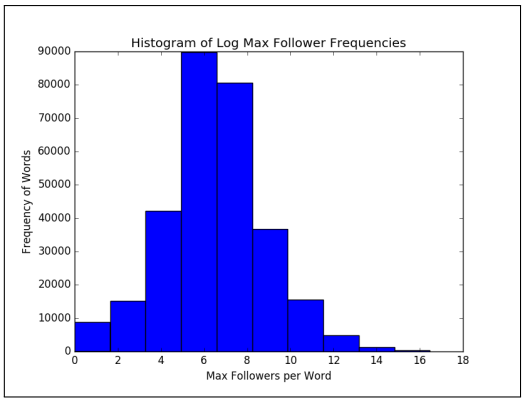
of this skew. We then analyzed the six diagrams to see how to standardize the data:



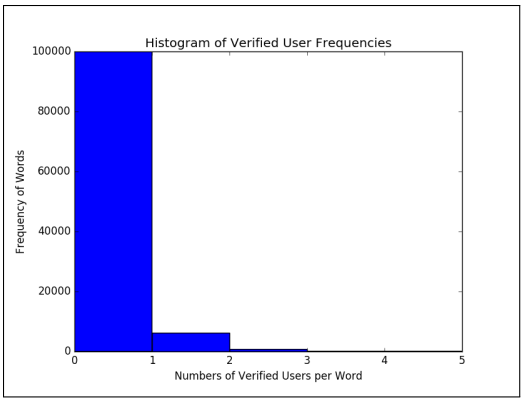


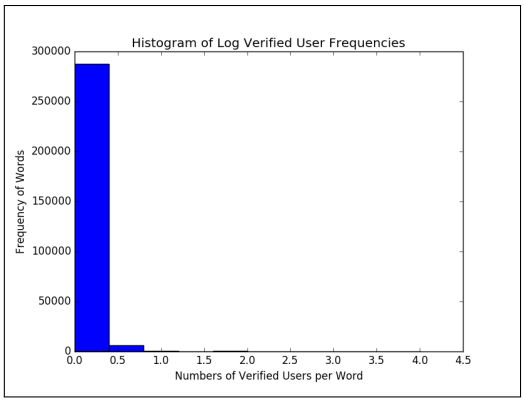
- *Term frequencies:* This distribution was still slightly skewed after two logarithm operations, but far less so than the original version. To normalize the values (and get rid of scale differentials between the three metrics), we took the z-scores of the term frequencies before including them in the algorithm. Since the distribution isn't perfectly normal, it's likely that items in the long right-skewed tail will be disproportionately weighted in the z-scores, but this isn't a problem (since our algorithm will likely want to pay particular attention to disproportionately high-volume tweets anyways).





- *Max Follower frequencies:* This distribution was almost perfectly normal after a logarithm operation, so we took the z-scores of all the values before including them in the final algorithm.





- *Verified frequencies:* This distribution, after one logarithm operation, had one large cluster at zero and another slightly higher. This is likely because most topics are tweeted by no verified users, while a few are tweeted by several. Even though this final distribution was skewed, we felt comfortable taking the z-scores of the values to normalize them – again because we wanted the algorithm to be particularly affected by items in the tail. While we considered using a chi-squared-type function to normalize the data (i.e., (observed frequency – mean)/mean), the fact that the mean was below one meant that the impact of this metric would be blown up.

After taking these normalizations, we had a collection of three numeric metrics and one Boolean (corresponding to whether the term was also in a hashtag or not). These four metrics all were associated with parameters and combined to form our completed authority-weighting algorithm.

$$\text{Rank By: } (\alpha * \text{freq}_{x,mod} + \beta * \text{follow}_{x,mod} + \gamma * \text{ver}_{x,mod})\delta$$

In order to determine the optimal values of the parameters, we began with numbers we thought seemed reasonable (1, 0.5, 0.5, and 1.5, respectively) based on how we thought the weighting should look, and manually tweaked each parameter to see whether our results would be improved. After such experimentation, we settled on our final parameters (1, 1.5, 1.5, 4) in our algorithm:

$$\text{Rank By: } (1 * \text{freq}_{x,mod} + 1.5 * \text{follow}_{x,mod} + 1.5 * \text{ver}_{x,mod})4$$

3.3.3. Results Over the course of one hour, we collected 200,000 tweets as our test sample. Running our algorithm on those tweets yielded the following results:

Rank	Prediction	Relevance
1	game	Slightly Relevant
2	police	Irrelevant
3	tuesday	Slightly Relevant
4	London	Irrelevant
5	traffic	Irrelevant
6	president	Slightly Relevant
7	trump	Highly Relevant
8	winner	Irrelevant
9	happy	Irrelevant
10	thanks	Irrelevant

3.3.4. Conclusion As expected, this authority-weighting algorithm was better at predicting trends than the count-based algorithm for text. Since it took into account factors that were likely to contribute to tweets moving into the trending list, we were able to capture soon-to-be trending topics that had not yet experienced spikes in popularity. However, given that we still had no past data to ignore tweets that had already trended or tweets that weren't growing in usage, the algorithm had a lot of false positives.

In order to improve this algorithm, a future iteration would have to create a more structured method of picking parameter values. Unfortunately, we didn't have the programming power to rapidly iterate through different possible parameter values based on what would lead to the most optimal rankings for evaluation. Furthermore, even if we developed an algorithm that could find the optimal parameters for a given data set, there's no guarantee that it would also find optimal parameters for any other sample – the best parameters would be those that were most likely to result in optimal rankings overall, which would be quite difficult to do computationally.

3.4. Method 4: Background Tweet Detection

3.4.1. Approach This method uses past tweet data to track topics that are being discussed at unusually high frequencies. Given that trends aren't necessarily topics discussed perpetually at high volumes, but are instead the topics that are rapidly rising in popularity or that are experiencing

temporary spikes in usage, tracking the usage of words in tweets against their baseline or normal usage could be a good way of predicting trends. To do this, we tracked the difference between the frequencies of tweets we saw in our test dataset and the corresponding frequencies in a baseline historical corpus.

$$\text{Rank by: } \left(\frac{\text{freq}_{curr} - \text{freq}_{hist}}{\text{freq}_{hist}} \right)^3$$

3.4.2. Implementation See commented code at Background Tracking Algorithm.

We began by collecting a large, historical corpus of tweets to serve as our 'baseline' for word usage. The topics that the Twitter world discusses change drastically depending on the time of day and day of the week, so we needed to standardize the collection of tweets as much as possible.[1] For example, if our baseline corpus was collected in the evenings but our sample test data was from the morning, we would likely find that terms like 'good morning' would get absurdly high scores given how 'abnormal' they would look. Thus, for the few days prior to our test collection, we collected tweets during the same 3-hour stretch from 3-6AM each morning, ultimately building a corpus of 1 million tweets.

Once we had our raw historical tweet data, we compiled word frequency data (in the same way we did for the previous algorithms). This time, we had to optimize the code further to ensure that the relevant data structures could hold all of the information we needed to collect. We ended up stripping the code of all unnecessary features and streamlining the opening/closing of files, as well as using only one dictionary to track overall counts to reduce the amount of memory the program would use (this way, the memory scaled directly with the total number of words across all the data sets, rather than the number of words in each set of 100,000 tweets, which was altogether about 8 times as large).

At this point, with the baseline values ready, we collected 200,000 tweets to serve as our test sample during a one-hour period from 5-6AM. After computing the word frequencies of the topics in these tweets, we then computed a score for each term based roughly on the difference between its

frequency in the sample timeframe and the baseline. By computing the chi-squared value of each term's divergence from the expected (baseline) frequency, we were able to avoid prioritizing large volume changes over small ones. Then, by cubing the results, we exaggerated the outcomes of this process to highlight the abnormally high divergences from the norms.

3.4.3. Results Over the course of several days, we collected a historical corpus of 1,000,000 tweets. We subsequently collected 200,000 tweets as our test sample over one hour. Running our algorithm on those tweets yielded the following results:

Rank	Prediction	Relevance
1	socceraid	Irrelevant
2	renato	Irrelevant
3	hummels	Highly Relevant
4	curry	Highly Relevant
5	tuesday	Slightly Relevant
6	bayern	Highly Relevant
7	munich	Highly Relevant
8	recountvp	Highly Relevant
9	aldubinitalyday2	Irrelevant
10	connected	Irrelevant

3.4.4. Conclusion This proved to be our best algorithm. First off, the fact that the program could differentiate between conventional high-volume tweets and unusual high-volume tweets automatically eliminated most of the false positives that had plagued the other systems. Secondly, the fact that the historical data was collected from the same time periods on previous days meant that we minimized the periodic variance of tweet topics that could have killed a background-tracking prediction method. Finally, the fact that we normalized the formula by dividing each term's score by its historically expected value meant that we could capture trends of both large and small magnitudes.

In order to improve on these results, we would have liked to have collected even more tweets over an even longer period of time (ie. an entire week) to check against variations in term frequencies over days of the week. How much data to collect is an interesting balancing act – on one hand, we don't want to track so much data that even terms like 'Christmas' won't be predicted to trend because they were popular last December, but at the same time we don't want to track so few tweets

that ‘Good Morning’ is predicted by our algorithm every day. Furthermore, it would have been interesting to explore questions around volume – for example should we have prioritized large scale differences over smaller scale ones?

3.5. Method 5: Short-Term Fluctuation Modeling

3.5.1. Approach This method uses changes in topic tweet volume over time to predict which phrases are most likely to become trending. By tracking the variation of each topic’s usage over a contiguous block of time, we theorized we’d be able to extrapolate which words had upward growth outlooks that suggested they’d soon be trending. We made this decision by calculating the differences in term frequencies during chunks of this time and by tracking whether or not these differences reflected upward movement in term frequencies:

$$\text{Rank by: } \left(\alpha\Delta_{5-4} + \beta\Delta_{4-3} + \gamma\Delta_{3-2} + \delta\Delta_{2-1}\right)$$

3.5.2. Implementation See commented code at Short-Term Fluctuation Algorithm.
In order to perform these calculations, we began by collecting the required data – a set of 500,000 tweets split into 100,000 tweet chunks, collected over a contiguous block of around 3 hours. We used topic frequency over the five chunks as the data points for our growth model, so we calculated the frequencies of each term for each of the time chunks independently.

Once we had stored the topic frequencies corresponding to each topic in each chunk of time, we then calculated the differences in frequency between contiguous blocks of time. These differences were scaled and then included in our final equation. Since we were aiming to extrapolate growth and predict behavior beyond the end of the data set, we weighted the changes in the later chunks higher than those in the initial ones. Thus, we ended up picking parameters of (0.4, 0.3, 0.2, and 0.1) for alpha, beta, gamma, and delta, respectively. That led us to our final equation:

$$\text{Rank by: } \left(0.4\Delta_{5-4} + 0.3\Delta_{4-3} + 0.2\Delta_{3-2} + 0.1\Delta_{2-1}\right)$$

3.5.3. Results Over the course of three hours, we collected 500,000 tweets as our test sample. Running our algorithm on those tweets yielded the following results:

Rank	Prediction	Relevance
1	mothers	Highly Relevant
2	happy	Highly Relevant
3	moms	Highly Relevant
4	transponder	Irrelevant
5	snail	Slightly Relevant
6	trecru	Irrelevant
7	found	Irrelevant
8	shipping	Irrelevant
9	trump	Highly Relevant
10	connected	Irrelevant

3.5.4. Conclusion Perhaps unsurprisingly, this algorithm was quite effective. Because trending topics are a rapidly-changing data set, we required an algorithm that was extremely responsive to marginal changes in topic usage over time – this method had the agility needed to highlight rising trends. Furthermore, strongly weighting those topics that were experiencing a snowball effect simulated the behavior of trends on Twitter quite well, and so this algorithm served as a great approximation for the actual metrics that Twitter values.

One way to build upon this success would be to create chunks that spanned shorter timeframes but that had greater volumes of tweets. That way, we would be able to track, with reasonable accuracy, the movements of tweets overall, while still remaining responsive to extremely small, temporary fluctuations, just as the Twitter algorithm does.

4. Recommendation Analysis

4.1. Approach

In building a recommendation system for trending topics, we followed the four-step process that engineers at Twitter themselves use. [2]

1. Generate Candidates – What are the possible items that the user might be interested in?
2. Filter – Which of these candidates are most likely to be appealing to the user?
3. Rank – In what order would the user want to see these topics?

4. Learn – Which items did the user like/dislike? What implications does that have for other recommendations?

We found that all of the predicted trending topics our algorithms generated were possible candidates for user interest. Since our use case was in giving users access to trends and conversations before they entered the mainstream dialectic, any early-detected trend could be interesting. That said, we then had to filter the actual recommendations down to a few, carefully curated options. Ultimately, we picked out those topics or hashtags that a user or his/her friends had previously tweeted, in the hopes that this would again be something of interest now that the discussion was picking up.[8] Our ranking was based on the likelihood of items making it onto the trending list going forward, which meant that we prioritized items that were higher on the list of predicted trends. Given that our task was to recommend trends to any user who called for them, rather than developing better recommendations over time for an individual user based on shared interactions, we didn't develop a learning apparatus for our algorithm.

4.2. Implementation

See code here: User Crawling Recommendation Engine

We implemented this approach by seeking out and obtaining both the user's past tweets and also those of his/her friends. After making the appropriate API calls and parsing through that text to find the potentially meaningful words/topics, we then proceeded down the list of items returned by our recommendation algorithm (in order, from 1-50). Every time we found a match between the two sets, we returned that match to the user. The algorithm aimed to return five recommendations, but would also sometimes return less if enough matches weren't found.

4.3. Conclusion

The basic version of our recommendation algorithm functioned well as an add-on to the prediction engine that filtered the right topics for users. Going forward, there are a number of ways to build on this – most centrally, by incorporating a feedback mechanism and a clustering algorithm that could push users towards clusters that they positively interacted with.

5. Evaluation

5.1. Prediction

To assess our results, we evaluated our prediction algorithms based on the quality of each predicted tweet—whether it actually became a trending tweet— within a series of evaluation metrics. For each list of 50 predicted tweet topics and hashtags, we compared the results with a list of actually trending tweets released in the hours after our predictions. We used this data to assess the relevance of our predicted pre-trending tweets to those that did trend. For each predicted tweet, we determine whether the topic or hashtag is actually on the trending list (highly relevant), relevant to the tweets on the trending list (slightly relevant), or completely irrelevant to the tweets on the trending list (irrelevant). See the code for collecting trending tweets here: Trend Collection Script

For each prediction method, we examined the following evaluation metrics:

- *Precision:* This metric expresses the proportion of the predicted topics and hashtags that actually become trending.

$$\frac{\text{\# of predictions that actually trend}}{\text{\# of predictions}}$$

- *Reciprocal Rank:* This measures how early in the predicted list the first trending topic/hashtag appears.

$$\frac{1}{\text{rank of highest ranking relevant prediction}}$$

- *Discounted Cumulative Gain (DCG):* This assesses the ranking quality of our predictions lists by measuring the gain of each predicted term based on its position in the list, with each lower result discounted.

$$\sum_{j=1}^{i} \frac{G(j)}{\log_2(1+j)}$$

  where $G(j)$ is the gain value of the j^{th} prediction.

- *Expected Reciprocal Rank (ERR):* This models the expected reciprocal length of time needed to

find a tweet topic or hashtag that actually trends.

$$\sum_{j=1}^{n} \left(\frac{1}{j} \prod_{k=1}^{j-1} (1 - R(score_k)) * R(score_j)\right)$$

where $R(score) = (1/2^{max})(2^{score} - 1)$ for scores $0, 1, \cdots, max$.

	Method 1	Method 2	Method 3	Method 4	Method 5
Precision	0.40	0.20	0.42	0.50	0.44
Reciprocal Rank	0.50	0.25	1.00	0.33	1.00
DCG	14.73	5.09	12.26	20.38	20.10
ERR	0.81	0.09	0.18	0.57	1.46

As shown in the evaluation table above, we found Method 4 to have the highest precision and discounted cumulative gain, thus proving to be the most accurate prediction method. The reciprocal rank and expected reciprocal rank of Method 4 were lower than many of the other methods, however, at 0.33 and 0.57, which suggests a less accurate *ranking order* despite high overall accuracy and relevance. Method 5 had a reciprocal rank of 1 and the highest expected reciprocal rank by far of 1.46, while maintaining the next highest precision and discounted cumulative gain after Method 4.

5.2. Recommendation

We performed qualitative surveys to see if the recommendations we provided were topical and of interest to real twitter-users. After seeing if the recommendations were adequate on Bharath's account, we used this algorithm on five other Twitter users. In all tests, at least one of the results was interesting or relevant to the test subject. While this is by no means an exhaustive survey, we took this success as a proof of concept for this method.

6. Conclusion

In this paper, we set out to investigate different algorithms that could detect soon-to-be-trending Twitter topics and recommend relevant ones to users. After evaluating each of our prediction methods, we found Methods 4 and 5 to be the most effective. While the first three methods

worked primarily with monitoring term frequencies, the last two methods accounted for more detailed frequency patterns over time and assessed different routine frequency trends to better detect abnormal frequency behaviors. Overall, Method 4 proved to be the most accurate and precise method of prediction analysis, yielding the most predictions that did ultimately trend. Method 5 came in a close second in terms of accuracy but far exceeded Method 4 in its reciprocal rank, thus suggesting a better ordering method that places more relevant predictions closer to the top of the predictions list. Furthermore, our recommendation engine served as a valuable addition to the analytics of the algorithms in our prediction section. By using past user tweet data to find potential topics of interest, it was able to provide value for all of our test subjects.

Going forward, we hope that these methods and algorithms can be refined to better help users take advantage of the powerful tool that is Twitter.

7. Future Work

7.1. Prediction Analysis

In addition to the five prediction methods implemented for this project, we recognize several additional limitations and considerations to be addressed in future work. Specifically, we developed and identified four other prediction algorithms.

7.1.1. Predict Pre-trending Tweets Based on Currently Trending Tweets After compiling a list of common tweets and hashtags, we can compare this list of possible hashtags with the list of currently trending hashtags. From here, we would weight the hashtags that are more similar to currently trending tweets more heavily than those that are more dissimilar.

We can thus use collaborative filtering to compare possible pre-trending tweets with currently trending tweets to predict accordingly. We would begin by representing each predicted and currently trending hashtag as a 0/1 vector, in which each hashtag is assigned 0s and 1s according to whether they satisfy given characteristics of words (e.g., prescribed word length, use of capitalization, single word, phrase, etc.) After creating 0/1 vectors for each of the resulting potential pre-trending tweets and the actual trending tweets, we would find the average vector for the currently trending tweets

to use as a benchmark for comparison against the possible pre-trending tweets. We then take each resulting hashtag found from our corpus of tweets and use both cosine similarity and Jaccard similarity to find the similarity between each possible pre-trending tweet vector and the average vector of actually trending tweets.

If we represent each possible pre-trending tweet vector as vector A_i and the average vector of currently trending tweets as vector B, we can find the cosine similarity between A_i and B for all i:

$$\cos(\theta) = \frac{A_i \cdot B}{||A_i||||B||}$$

Similarly, we can find the Jaccard similarity between A_i and B for all i:

$$J(A_i, B) = \frac{|A_i \cap B|}{|A_i \cup B|}$$

Given that we are using the average vector across all 0/1 vectors of the currently trending tweets, it is likely that vector B will not be a 0/1 vector, but rather made up of fractions between 0 and 1 that result from averaging all of the different vectors. In order to better account for the actual values in vector B, we would likely work primarily with cosine similarity over Jaccard similarity, since cosine similarity takes into account the lengths of each vector.

In addition to assessing similarity based on word and hashtag characteristics, we would also use WordNet to compare similarity in semantics between the possible pre-trending tweets and currently trending tweets. Finally, we would weight the possible pre-trending tweets that are more similar to the currently trending tweets, both in characteristics and semantics, more heavily in predicting which hashtags and topics would likely trend. We found that a WordNet similarity analysis extended beyond the scope of this class and thus did not implement this method of predicting tweets for this project.

7.1.2. Cross-check Possible Pre-trending Hashtags with Other Tweets for Relevance After compiling a list of commonly found hashtags and topics in our corpus of tweets, we can then compare the resulting hashtags with other posts to find semantic similarity. With this method, the

goal is to identify commonly used hashtags that could be applied to more tweets in the corpus than those already included. For example, if #government were a commonly found hashtag, and then the corpus also had other tweets related to government issues with #election and #president, but not #government, we could conclude that #government was in fact more applicable and relevant than initially found by purely counting the number of posts with #government explicitly written. From here, we would weight #government more heavily to account for its wider relevance and applicability.

This method would also require Wordnet similarity analysis to determine which tweets would be relevant to different hashtags and topics. Again, we find this to extend beyond the scope of this class for the purposes of this project.

7.1.3. Predict Trends with Machine Learning Another possible method of predicting Twitter trends involves using various machine learning algorithms to train a dataset of tweets to predict new trending tweets. With this method, we would start by collecting 200 words (hashtags or topics) that did trend on Twitter and 200 words that did not trend. We would then use this information to train our dataset of real-time tweets using various machine learning algorithms.

In deciding which machine learning algorithms to implement, we could explore various classification models, including Naive Bayes, decision trees, k-nearest neighbors, support vector machines, or logistic regressions. We believe k-nearest neighbors classification to be a particularly promising method, and as a model, we study one specific machine learning algorithm developed by Associate Professor at MIT Devavrat Shah (mentioned earlier) that was found to be extremely effective in predicting Twitter trends. Professor Shah turns away from purely parametric models of predicting trends to develop a data-driven approach, in which he compares each new topic, denoted as observation s, to example patterns of activity r of previously trending topics [5]. In this approach, Shah takes a vote $V(\mathrm{r}, \mathrm{s})$ of whether the topic trends or not based on the similarity between r and s. This similarity is determined based on the Euclidean distance $d(\mathrm{r}, \mathrm{s})$ between the example pattern and observation and weighted according to a decaying exponential model [5]:

$$V(\mathrm{r}, \mathrm{s}) = e^{-\lambda d(\mathrm{r}, \mathrm{s})}$$

where λ denotes the "sphere of influence" of each example to scale accordingly [5].

Finally, Shah's algorithm sums all of the trending votes $R+$ and non-trending votes $R-$ to determine whether the the ratio of the sums is greater than or less than 1, where ratios greater than 1 denote observations that are likely to trend [5]:

$$\sum_{r \in R+} e^{-\lambda d(\mathrm{r}, \mathrm{s})} / \sum_{r \in R-} e^{-\lambda d(\mathrm{r}, \mathrm{s})} > 1$$

Using a machine learning algorithm like Shah's would then lead us to produce a new dataset of tweet topics and hashtags that are likely to trend. We found that the machine learning algorithms involved in this prediction method extended beyond the scope of the material covered in this course and thus did not implement this method in the time provided for this project.

7.1.4. Track Tweet Frequencies Based on Previous Time Patterns In determining trending tweets, it is important to not only note tweets with high frequencies, but also to identify tweets that have abnormally high frequencies at a given time. We thus consider another method that detects certain words and topics that may tend to have higher spikes in frequency at certain times on a regular basis. For example, "rush hour" may be a common topic on weekdays around 8-10 am and 4-6 pm, during which times the phrase would show higher spikes in posting on Twitter. Given this frequency pattern, finding high counts of tweets including "rush hour" during the phrase's common peak times would not be abnormal and thus not trending. To account for such cases, we consider two methods of prediction analysis:

Track Frequencies across Multiple Dimensions

Following this method, we identify the time of a spike in frequency for a given tweet topic or hashtag and examine the frequencies of this term across different dimensions of the given time. For example, given the example of "rush hour," in which we find a spike in the usage of "rush hour" at 9 am on a Monday, we would then find the frequencies of "rush hour"

around 9 am every day and the frequencies of "rush hour" every Monday for the past month to identify any patterns in frequencies. From here, we could then determine whether the spike in frequency is abnormal and thus trending.

Graph Frequency Trends over Time

Here, we graph the frequencies of each term over time to plot the frequency trends for each possible tweet topic or hashtag. For example, given the example of "rush hour", we would likely find a plot of frequencies that stays fairly constant at a low rate with large spikes from 8-10 am and 4-6 pm on weekdays. From here, we can compare instances of high frequencies for a given term with the plotted frequency trends to determine whether there are abnormal peaks. Any spikes in frequency that correspond with routine peaks would thus not qualify as trending.

While these methods of prediction analysis could improve our results and prediction accuracy, they would be considerably more involved than our implemented methods and require more time for analysis and computer processing power than we had access to for the scope and purposes of this project. Although we were unable to implement this next step of analysis, we recognize its power and potential for future work.

7.1.5. Additional Considerations We also recognize additional limitations with our implemented methods. To begin, our current methods identify individual terms and single words but do not account for non-unigram terms. For example, given a celebrity like "Michael Jackson," our current prediction algorithms would classify "Michael" and "Jackson" separately and thus skew results. More advanced machine learning techniques would be required to account for such instances and improve our predictions accordingly.

Another consideration to take into account for future work is the effect of misspelled words. In our implemented methods, different spellings of the same word would be classified as different words. As a result, if a commonly used word is misspelled in certain tweets, the relevance and trending potential of that word or topic would be underrepresented under our current models. With more time, this problem could be addressed by checking for common misspellings of popular terms

to group together and comparing words to each other based on similarities in letter sequencing such that similarly spelled words are checked for overlap.

Furthermore, we recognize the potential use of WordNet to check for differentiation in word tenses or plurality. As with many other considered methods of prediction analysis, we leave this potential algorithm enhancement for future work beyond the scope of this course.

7.2. Recommendation Analysis

First, a deeper understanding of Natural Language Processing (NLP) or WordNets could allow us to draw valuable insights from particular words to other words. For example, one possible algorithm would recommend not just topics previously tweeted about by a given user, but also topics that are similar in context to the user's interests or past activity, based on some calculated similarity between words. Another way to implement this proposal would have been to develop a clustering algorithm for all terms based on the tweets that they found themselves in and the words that they were in close proximity to. Then, we could simply assign users to certain clusters based on their past tweet history, and draw recommendations from there.

Second, explicit preference ranking on the part of the user or their friends would have allowed us to use techniques like collaborative filtering in picking out more relevant topics going forward. With such a system, we would have been able to rank soon-to-be trending words for users based on no information about the words themselves.

8. Honor Code

This paper represents my own work in accordance with University Regulations. – Bharath Srivatsan and Kelly Zhou

References

[1] Lecture 6 - Trend Detection In Twitter Social Data (Analyzing Big Data With Twitter), Berkeley School of Information.

[2] Lecture 12 -Analyzing Big Data with Twitter: Recommender Systems by Alpa Jain, Berkeley School of Information.

[3] "Stopword list," in *RANKS NL*.

[4] "Twitter api wrapper," in *tweepy*, 2009.

[5] "Early detection of twitter trends explained," in *Computational Amusement*, 2012.

[6] "Get statuses/sample," in *Bad Hessian*, 2012.
[7] "Social networking is the no. 1 activity in the u.s." in *Statista*, 2013.
[8] "Rate limits: Chart," in *Twitter, Inc.*, 2016.
[9] "Streaming api request parameters," in *Twitter, Inc.*, 2016.
[10] A. L. Evan Miller, Kiran Vodrahalli, "Estimating trending topics on twitter with small subsets of the total data," 2015.
[11] L. Grossman, "The 500 most frequently used words on twitter," in *TIME*, 2009.
[12] R. Laraway, "New insights on how followers benefit small and medium-sized businesses," in *Twitter, Inc.*, 2014.
[13] Z. X. Rong Lu and Q. Yang, "Trends predicting of topics on twitter based on macd," in *National Laboratory of Pattern Recognition*, 2012.
[14] E.-P. L. Su Mon Kywe and F. Zhu, "A survey of recommender systems in twitter," in *Social Informatics: 4th International Conference, SocInfo 2012, Lausanne, Switzerland, December 5-7, 2012. Proceedings. , 7710 , 420. Research Collection School Of Information Systems.*, 2012.
[15] Y. W. Tianxi Li and Y. Zhang, "Twitter hash tag prediction algorithm," 2011.