CLASSIFICATION OF TWEETS VIA CLUSTERING OF HASHTAGS

EECS 545 FINAL PROJECT, FALL, 2011

DOLAN ANTENUCCI, GREGORY HANDY, AKSHAY MODI, MILLER TINKERHESS

Abstract. We present two techniques to aid in the retrieval of information from Twitter. First we present a technique to cluster hashtags in meaningful topic groups using a combination of co-occurrence frequency, graph clustering and textual similarity. Second, we present a technique to classify a tweet in terms of these topic groups based on their word content using a combination of PCA dimensionality reduction and a variety of multi-class classification algorithms. We examine the relationship between the clustering step and the classification step to evaluate the performance of each.

Keywords: Twitter, hashtag, machine learning, clustering, classification

1. Introduction

Twitter is the leading microblogging social network. It is the ninth most popular site on the Internet with over 200 million registered users producing over 200 million tweets every day [1, 2, 3]. Users post publicly viewable tweets of up to 140 characters in length, and follow other users whose tweets they are interested in receiving. The sheer volume of data produced by Twitter makes it an attractive area of study for machine learning. Unfortunately, many standard algorithms for extracting information from a body of text assume correct English. As a result, they are ineffective at analyzing tweets, which often contain slang, acronyms, or incorrect spelling or grammar.

Some words within a tweet are prefixed with punctuation symbols to indicate special meaning. For example, a word prefixed by "#" is a *hashtag*. Hashtags are a way for a user to indicate the subject of a tweet in a way that is easy to search for; hashtags are deliberate metadata. We make the simplifying assumption that a tweet's hashtag content is a good approximation of its total content [4]. It follows that when multiple hashtags occur in the same tweet, they represent a similar approximation of content; the more often two hashtags co-occur, the more similar their meanings are.

In this paper we present an algorithm to learn the relationships between the literal content of a tweet and the types of hashtags that could accurately describe that content. As a classification problem, each tweet is represented as a frequency list of the non-stopword, non-hashtag words that appear in the tweet; its categories are the hashtags that are used in the tweet. In order to successfully classify tweets, we overcome problems of co-occurrence based graph clustering, dimensionality reduction, and multi-class categorization. We show

Date: December 16, 2011.

that clustering algorithms and dimensionality reduction allow us to perform supervised classification on an otherwise intractable problem; that textual similarity measures may be used to expand graph-based clustering techniques without a resulting loss of precision; and that multi-class classification performs better than naive approaches on heavily compressed data.

Hashtag assignment could be used to suggest a tag to a user while they are composing a tweet, or to categorize untagged tweets as in semi-supervised learning. Hashtag clustering could also be used independently of classification, for example by inferring a user's topics of interest from the clusters to which their most frequently used hashtags belong; users could be shown other tweets, user profiles or advertisements that correspond to those interests.

2. Related Work

In his work on clustering hashtags, Poschko argues that two hashtags are similar if they co-occur in a tweet[5]. He creates a clustered graph with co-occurrence frequency as the distance measure. We expand upon his work by introducing a novel method for measuring the similarity between two hashtags. We use a larger set of hashtags and test several clustering methods instead of focusing on only one.

Other work on clustering text-related entities typically focuses on a bag-of-words model that takes all the words of the entity followed by dimensionality reduction to make clustering computationally feasible [6, 7]. We perform a similar dimensionality reduction to make classification feasible.

Davidov et al. attempt to classify the sentiment of a tweet in terms of its mood or opinion [8]. This requires a considerable amount of linguistic analysis. Our focus on classifying the topic of the tweet instead of the sentiment will not require this analysis, providing a simpler tool for classification.

Mazzia and Juett investigate the recommendation of hashtags based on a tweet's content using naive Bayes classification [9]. We expand on this work by classifying a tweet into broader hashtag-cluster categories.

3. Our Approach

3.1. **Data Description.** Our data set is a collection of 476 million tweets from 17 million unique users spanning from June 2009 to December 2009 collected by Stanford University through their partnership with Twitter. We limit this data set to 178 million tweets during July and August in order to save on computation costs for our clustering and classification.

We first perform several preprocessing steps to prepare the data for clustering. We remove all tweets containing non-ASCII characters in order to focus on English-language content. From the resulting set, we remove the tweets that contain no hashtags, which leaves around 16 million tweets. Following Rosa et al., we remove all non-alphanumeric characters, resulting in a list of hashtags and non-hashtag *terms* (i.e. words, numbers) used in each tweet. The resulting data set contains 872,402 hashtags [4].

It may seem worrisome that less than ten percent of the tweets in our subset contain a hashtag, and that we are basing much of our learning on the presence of hashtags. However, we still have more than enough data to detect patterns in. Furthermore, we see this as a potential application area for semi-supervised learning. Once a reliable method of predicting

the hashtag content of a tweet is developed, the remaining 90 percent of tweets could be assigned hashtag labels to summarize their subject matter.

The final preprocessing step is to remove certain hashtags that are either auto-generated by external systems or are *overly-general* hashtags, i.e. those whose usage is too broad to consider their co-occurence with other tags to be meaningful. For example, the hashtag $\#fb$ is automatically appended by a particular Facebook application that shares Facebook posts on Twitter. An example of an overly-general hashtag is *#followfriday*, which is a hashtag used on Fridays to recommend other users to follow.

From this preprocessed data, we generate a list of the most frequently used hashtags, along with the co-occurrence frequencies of all hashtags with each other. We limit our data set to the most frequently occurring hashtags and the hashtags that they co-occur with in order to simplify the clustering problem.

3.2. **Clustering Hashtags.**

3.2.1. *Difficulties of Clustering.* There are a number of variables that make it difficult to cluster hashtags. The first major difficulty is that hashtags are not required to be well-defined words. For example, the hashtag $\#p2$ is often used in tweets with politically progressive content. In many cases, hashtags are a concatenation of words like *#iamthemob*.

Some similarity measures like the Wu-Palmer distance [10] and path distance are based on the synonyms of the words being examined, which means that they can't be used as the central component in a clustering algorithm such as k-means. However, our algorithm will still use this concept in a more limited scope.

The fact that hashtags are often concatenations of words makes it difficult to apply the Levenshtein distance [11] or other edit distances. For example, under edit distance the hashtags *#iamblessed* and *#iamthemob* are close to one another when in fact their semantic meanings are very different. We tried to implement a method using this distance, but it resulted in poor results.

The set of hashtags is constantly increasing in size, with new, "trendy" hashtags appearing everyday. Therefore we want to use an algorithm that captures the essence of the most-used hashtags, while remaining computationally tractable. We decided to focus clustering on the 2,000 most frequently used hashtags. However, our algorithm does not discard the lesser-used hashtags; instead, it focuses its efforts on the most-used hashtags.

3.2.2. *Co-Occurrence Relation.* Poschko proposes that two hashtags are similar if they co-occur in a tweet [5]. Two hashtags may be considered similar to each other if they appear in tweets that discuss the same topic, and are therefore even more similar if they appear within the same tweet. Poschko calculates the Wu-Palmer distance between hashtags that co-occur and shows that this value is higher than the distance between two randomly chosen words. Although the Wu-Palmer distance is a poor distance for hashtags as a whole because they often are not single words, it does provide a good measure over the set of hashtags only containing single words. We propose that because the Wu-Palmer distance is high for reoccurring hashtags that are real words, then the set of all reoccurring hashtags can be used as baseline for a similarity measure.

After creating the co-occurrence lists for the 2,000 most used hashtags, we used the Natural Language Toolkit (NLTK) Python library to find the Wu-Palmer distance for each hashtag in each list, then took the average over all of the lists. For hashtags that co-occur, this value was .40, and for random words it was .16. These results verify that the claim by Poschko applies to our data set: that the hashtags found in the co-occurrence list are much more similar to each other than random words are.

3.2.3. *Minimizing the Level of Noise.* Even though the list as a whole exhibits similarity, there is still noise in the data. For example, *#photography* is a popular hashtag used to denote a photograph, and another hashtag is often used within the same tweet to describe the place the picture was taken, such as *#iran*. In this case, the fact that these hashtags co-occur does not mean that they are similar by our definition of similarity. To help minimize this problem, let n_{ij} be the total number of co-occurrences between hashtag i and hashtag j. Hashtags A and B are kept on the co-occurring list if and only if

$$min(\frac{n_{AB}}{\sum n_{Aj}}, \frac{n_{BA}}{\sum n_{Bj}}) > 0.05.$$

This means that hashtags A and B account for at least 5% of each others co-occurring hashtags. Several different percentages were tested and this one provided the best results.

Noisy relationships still exists after this step. Therefore, we perform another round of filtering by comparing the contents of the hashtags' respective co-occurrence lists. Let m be the total number of hashtags that occur in both hashtag A's and hashtag B's co-occurrence lists. Further, let m_A be the total number of occurrences of these hashtags in list A, and m_B be the total number of occurrences in list B. Then the relationship between hashtag A and hashtag B is maintained if an only if

$$min(\frac{m_A}{\sum n_{Aj}}, \frac{m_B}{\sum n_{Bj}}) > 0.2.$$

Note that by the first level of filtering, this minimum is at least 0.05. Comparing each list takes a considerable amount of time to run, and this bottleneck was a motivating factor in our decision to constrain our focus to only 2,000 hashtags. However, it is easily parallelizable, and when programmed efficiently could allow a much greater number of hashtags to be considered; we leave such optimizations for future work.

3.2.4. *Defining the Similarity Measure.* We define the following similarity measure between hashtags A and hashtags B:

$$S(A, B) = (\frac{n_{AB}}{\sum n_{Aj}} + \frac{n_{BA}}{\sum n_{Bj}})/2.$$

It should be noted that several similarity measures were tested, such as multiplying the two ratios together, but this proved to be the best measure.

3.2.5. *Spectral Clustering and METIS.* With our well-defined similarity measure, and therefore a similarity matrix, spectral clustering was the ideal candidate to perform clustering over our list of hashtags. For comparison purposes, we performed spectral clustering and normalized spectral clustering. We also examined the graph program METIS that claimed to perform 10% to 50% better than spectral algorithms [12]. This algorithm is based on

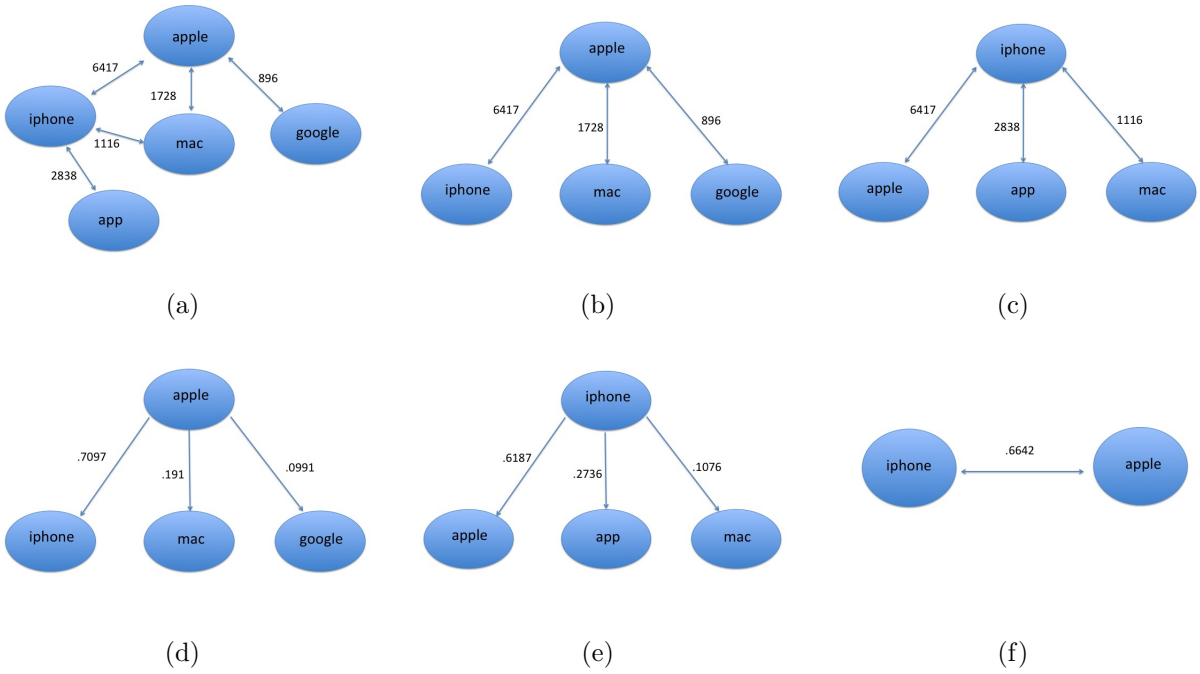

FIGURE 1. Subfigure (a) represents the original undirected graph proposed by the co-occurring lists. Subfigures (b) and (c) consist only of the neighbors of apple and iphone respectively (the graph remains undirected). Subfigures (d) and (e) convert the undirected graphs into an directed graph by weighing the edges. The filter process examines these directed graphs, and after passing the final undirected graph with the similarity measure is created (f).

multilevel recursive-bisection and multilevel k-way partitioning schemes. Instead of using an Laplacian matrix like the spectral methods, METIS takes in the similarity matrix, and uses the similarity measures to define a weighted graph.

Even though these methods cluster over the 2,000 most used hashtags, we maintain the co-occurring lists with each of these hashtags. As a result, even the hashtags which are not in the top 2,000 are then assigned to a cluster. Referring back to Figure 1, if apple and iphone are clustered together, then mac, google, and app will appear in the same cluster. If a hashtag appears in more than one cluster, it is assigned to the one it occurs most frequently in. In all, 26,028 hashtags are clustered using the spectral methods and 26,006 are clustered using METIS.

3.2.6. *Expanding Clusters with Cosine Similarity Measure.* In order to expand the breadth of our clustering techniques, we optionally find unclustered tags that are textually very similar to a tag in an existing cluster, and then expand the cluster to include the previously-unclassified tag.

To determine the textual similarity of two tags, we first vectorize the tags and then take their cosine similarity. Our vectorization technique is similar to taking the character-wise

trigram of a word, with a simplification in order to reduce the cost of computing the cosine of two vectors.

We represent each tag t as a list of tokens with one token for each letter in the tag, plus an additional dummy token at the beginning of the list and one at the end of the list. The binary high-dimensional sparse representation b of a tag, indexed using $i, j, k \in [1 - 38]$, is

$$b(tag)_{i,j,k} = 1_{\{subseq(i,j,k) \in tag\}}$$

In other words, each dimension in b corresponds to a possible sequence of three characters; the corresponding value in b is 1 if that sequence of characters is in the list of tokens l, or 0 otherwise. Restricting these values to 0 or 1 allows us to find the dot product of two vectors more quickly than allowing each dimension to take on larger range of values. The cosine similarity between two binary vectors is the dot product of the two vectors divided by the product of the magnitudes of the vectors:

$$cos(x, y) = \frac{\sum_{i,j,k} 1_{\{x_{i,j,k}=y_{i,j,k}=1\}}}{\sqrt{\sum_j x_j^2} + \sqrt{\sum_j y_j^2}}$$

In order to pair an unclassified tag with an existing cluster, we find the clustered tag from the top 2,000 tags that it is most similar to. If their similarity is above the threshold 0.5, we add the tag to the cluster. Otherwise, we leave the tag unclustered. Performing this additional hashtag-class assignment increases the number of tags assigned to any cluster from 26,028 to 52,146 under the spectral clustering methods, and from 26,006 to 52,032 under METIS.

3.3. **Classification.** In the classification step, we attempt to classify tweets into classes defined by the hashtag clusters. Intuitively, the idea is that given the text of a tweet, the classification algorithms will be able to suggest what hashtags it should contain. Since we have actually clustered the hashtags, we will not be suggesting what hashtags the tweet will contain, but will suggest which cluster the hashtag is most likely to fall into.

3.3.1. *Description of feature vectors and classes.* For classification, we have a corpus of text tweets that we will map to clusters of hashtags. We can consider each tweet a document and use standard document classification techniques in this step. We create a set of all unique words (removing common stopwords and all hashtags) that occur in all the tweets. Let us assume that there are d words in the dictionary. The feature vector is a d-dimensional integer-valued vector where the i^{th} entry in the vector represents the frequency in the tweet of the i^{th} word in the set.

The classes that each feature vector is classified into is represented by the hashtag clusters. A tweet belongs to a class if a hashtag appearing in the tweet is part of a certain cluster. When generating the training data, if a tweet has multiple hashtags, we consider that tweet to be a part of multiple classes so we add the same training point multiple times with different classes as the target. We do this since the text of a tweet is related to all the hashtags in the tweet.

Due to the size of our training sets, each feature vector was an extremely-sparse high-dimensional vector. Due to the number of training samples and the high dimensionality of

the vector, attempting to classify over the complete data was intractable on any machines that we had access to. To get around this problem, we used two solutions.

3.3.2. *Classification using Dimensionality Reduction.* We used the PCA tools from the Python library*scikit-learn* to reduce the dimensionality in our data. For our data set containing 100,000 tweets, we had approximately 46,000 dimensions. We reduced this to 100 dimensions using PCA. Once we reduced the number of dimensions, it was possible to apply standard classification algorithms to the data. We applied naive Bayes and Linear Discriminant Analysis the to the reduced-dimension data.

Naive Bayes initially performed poorly because PCA creates real-valued feature vectors whose values don't always mean much. Instead, we used scikit-learn's implementation of Bernoulli naive Bayes, which binarizes the data by replacing all positive values in the feature vector with the value 1 and all negative values with the value 0. This performed much better than the other naive Bayes implementations. LDA performed acceptably well on the dimension-reduced data without the need to create a binary representation of the data.

3.3.3. *Classification using SVM algorithm optimized for sparse vectors.* An alternative approach to deal with the high dimensionality of our feature vectors is to use an algorithm that was designed to use the sparse representation of a vector. By representing the feature vectors as a sparse vector, the problem becomes tractable and we don't need to do any dimensionality reduction. We used scikit-learn's implementation of a radial basis SVM classifier, which supports sparse data represenatations. We use a one-versus-all strategy to make SVM a multi-class classifier. We also perform SVM classification on the PCA-reduced data.

3.3.4. *Majority Vote classification.* We determine the majority vote classification accuracy to use as a baseline measure. The majority vote accuracy is defined as the percentage of data points whose class label is the most common class label.

4. Experiments and Results

4.1. **Clustering.** We focused on clustering the top 2,000 most used hashtags. Decreasing this number would reduce the breadth of our clusters coverage; on the other hand, increasing this number made the similiarity filtering step intractable. We performed several trials on the 5,000 most frequently used hashtags without running the step, but most of the hashtags ended up in one large cluster while the rest of the clusters consisted of single hashtags. We believe that this was caused by noisy connections, which inadvertently linked most of the hashtags closely together. This issue is removed during the filtering described earlier.

Only 1,557 hashtags remained after filtering. Unfortunately, there is no intuitive way to pick the correct number of clusters to generate. Spectral clustering does use k-means on the eigenvectors of the Laplacian matrix, which suggests that we could graph the within-cluster scatter and find the knee. Unfortunately, for spectral clustering, increasing k increases the dimension of the eigenvectors that are evaluated in k-means. As a result, an increase in k leads to an increase in within-cluster scatter. We tested several different values for k, and evaluated the size of the clusters that were formed. A small value for k resulted in hashtag clusters that were not indicative of one single topic. Picking a large value of k separated

clusters that should actually be in the same group. We decided that k=300 provided the best trade-off between these two problems.

In order to evaluate the clustering techniques we randomly selected 100 of the 300 clusters and score them manually on a scale of 1 to 5, with 1 representing a cluster that makes no sense and 5 being a perfect cluster. Factors that go into this rating include size of the cluster, number of matching hashtags that relate to the topics, and relation of topics if more than one topic arises. Some examples of clusters and their ratings are given below:

```
bears beatles chargers cowboys fantasyfootball jets nyg packers panthers
patriots raiders steelers vick vikings
Score: 5 (collection of all football teams)
amazing comic comics cool funny humor strange voss webcomic webcomics weird
Score: 3 (clearly comics are represented, but there are random words such as
amazing and weird)
crnc failedchildrensbooktitles fishy grizzly obamacare waterloo
Score: 1 (list looks simply random)
```

TABLE 1. Ratings for the Clusters

Method	1	2	3	4	5	Average
Spectral	21	6	10	21	35	3.23
Normalized Spectral	36	7	8	15	34	3.04
METIS	30	13	13	13	15	2.22

It is clear from Table 1 that spectral clusters performs the best, with normalized spectral close behind. Notably, normalized spectral tended to score around the two extremes, with not many scores in the middle values. METIS contained many clusters that should have been clustered together. For example, it contained five clusters of baseball teams, whereas the spectral methods grouped these teams into the same cluster. This implies that the clusters created by the spectral method would be more useful in identifying topics comprising may hashtags. It's also possible that METIS would avoid this issue if k was chosen to be a smaller number.

4.2. **Classification.** We split our data into 90% training data and 10% testing data. We do not perform cross-validation; however, anecdotally, results did not vary by more than a couple of percentage point over several different divisions of the data.

We use majority-vote classification accuracy as a baseline to measure our performance against. The majority-vote accuracy is simply the percentage of the data points whose cluster is the most common cluster among the data points.

4.3. **Discussion.** Over all clustering techniques, all classification techniques using PCA-reduced data perform better than majority vote; SVM without PCA reduction performs no better than majority vote. This shows that dimensionality reduction serves two purposes: it makes the problem computationally feasible, and it improves performance by capturing regularities in the data.

TABLE 2. Classification Performance

	Bern. N.B.	LDA	SVM	Majority	SVM, no PCA
Spectral	26.5	18.3	25.1	11.7	11.8
Spectral w/ Cos. Sim.	26.5	18.6	24.1	11.1	10.8
Norm. Spectral	21.4	17.2	25.2	17.9	17.9
Norm. Spectral w/ Cos. Sim.	22.4	17.3	25.1	17.4	17.6
METIS	14.8	11.8	13.8	4.1	4.1
METIS w/ Cos. Sim.	14.9	12.1	14.1	3.9	3.9

Our data show that the classification techniques performed similarly well when given clusters that included hashtags added via cosine-similarity measures as when they did not. This suggests that the cosine similarity technique allows us to expand the breadth of clustering without negatively affecting the semantic cohesiveness of the clusters.

Although all three classification techniques over PCA-reduced data perform better than majority vote, the Bernoulli Naive Bayes and SVM classifiers perform better than LDA over all clustering techniques. This suggests that when PCA reduction is applied the data points are not easily linearly separable, but that limitation may be overcome in the case of Bernoulli Naive Bayes due to the binarization of the data, or in the case of the SVM classifier by using a non-linear kernel.

Under all classification techniques, data clustered by the METIS algorithm had a lower classification success rate than all other clustering algorithms. In our manual examination of the clusters produced by each technique, we saw that METIS produces clusters that seem less semantically cohesive than the other techniques. We conjecture that classification accuracy is therefore an indicator of the quality of a clustering technique. Alternatively, METIS's worse performance may be a result of the distribution of tags to clusters; the lower accuracy of the majority vote classifier indicates that hashtags are more evenly spread among clusters under METIS than under the other techniques, which may result in more difficulty performing classification regardless of the semantic cohesiveness of the clusters.

5. Future Work

5.1. **Clustering.** We have shown that our clustering method has promise; we believe the method could be further improved. During the preprocessing step illustrated in Figure 1 the undirected graph is transformed to a directed graph that has the properties of a Markov chain. It would therefore be possible to perform a Markov Cluster Algorithm as proposed by Dongen [13]. It is possible that this algorithm would be less affected by noise, which would reduce the need for the preprocessing step and allow a larger number of hashtags to be clustered. Parallelizing the preprocessing step could also allow us to cluster a larger number of hashtags.

There may be a better way to pick the number of clusters. Increasing K was meant to increase the number of topics the clusters represented. However, instead of splitting large clusters apart into smaller topics, our algorithm would often pull one or two hashtags out at a time. This was most likely because k-means was always done on the full graph. It could be

better to first apply k-means to the whole set, and have k be a small number. Then for each of these clusters, perform localized k-means. This method could avoid the issue of creating clusters that are too small in size.

5.2. **Classification.** We discovered that the performance of classification algorithms improves vastly if we shorten the time frame from which the data is drawn. It would be worth exploring the effects of introducing temporal information into our feature vectors since the importance of some hashtags seems to be limited to specific times.

Extending the co-occurrence idea from clustering, it could be worth exploring clustering of non-hashtag terms using a similarly defined measure and performing classification using these clusters of terms. This would result in lower-dimensional feature vectors, possibly making learning algorithms tractable without the use of PCA. Using this co-occurrence data might also be useful in classification of documents from different domains.

6. Conclusion

With a growing, diverse membership and masses of new information being added daily [14], opportunities to find useful information from Twitter will also grow. We have presented two new approaches that help unlock this information. The first focuses on clustering hashtags into meaningful topics; the second uses these clusters to identify the topic of tweets through classification. We also present two novel similarity measures that could be useful beyond the scope of the methods we examined.

While we are pleased with the preliminary findings of our clustering and classification methods, our project hit some bumps along the way, causing us to deviate from our initial path. At first we attempted to cluster hashtags based on string similarity instead of semantic similarity. We found that the clusters created using these metrics were not very meaningful. This makes sense in hindsight becuase there are no rules of hashtag usage–they are used in a diverse variety of ways, and there are many unique hashtags that are semantically similar but not structurally similar. We also initially believed we would be able predict what a certain user is going to tweet about next based on their previously used hashtags. Instead, we found that many users preferred a very small set of unique hashtags, which made prediction more reliable but much less meaningful. These roadblocks led us into our current project plan, which ended up being a promising direction for future work.

7. Project Member Contributions

All team members participated in data processing early in the project. Most theory behind our project was a team effort. Large scale data processing was handled by Dolan. Our preliminary research into hashtag usage prediction was co-developed by Akshay and Miller, working on naive Bayes and LDA, respectively. Classifying hashtags based on tweets was primarily coded by Akshay and Miller, with Dolan assisting with execution. Exploration into and coding of different clustering techniques, and the development of the similarity measure and filtering was handled by Greg. Clustering analysis was done by Dolan and Greg. Miller worked on the trigram cosine similarity metric. Writing the paper was a team effort.

References

[1] M. Shiels. Twitter co-founder jack dorsey rejoins company. *BBC News*, 2011.

[2] Alexa. Alexa top 500 sites on the web, 2011. URL http://www.alexa.com/topsites.

[3] Twitter.com. Your World, More Connected, 2011. URL http://blog.twitter.com/2011/08/your-world-more-connected.html.

[4] Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. Topical Clustering of Tweets. *Corpus*, 2011. URL http://www.cs.cmu.edu/~kdelaros/sigir-swsm-2011.pdf.

[5] Jan Poschko. Exploring twitter hashtags. Retrieved from arXiv.org; not yet published, 2011. URL http://arxiv.org/pdf/1111.6553v1.

[6] Anand Karandikar. Clustering short status messages : A topic model based approach. *Work*, 2010. URL http://ebiquity.umbc.edu/get/a/publication/518.pdf.

[7] Marc Cheong and Vincent Lee. *A Study on Detecting Patterns in Twitter Intra-topic User and Message Clustering*. IEEE, August 2010. ISBN 978-1-4244-7542-1. 3125–3128 pp. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5597282.

[8] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. *staffscienceuvanl*, (August):241–249, 2010. URL http://eprints.pascal-network.org/archive/00007073/.

[9] Allie Mazzia and James Juett. Suggesting hashtags on twitter. EECS 545 Project, Winter Term, 2011. URL http://www-personal.umich.edu/~amazzia/pubs/545-final.pdf.

[10] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, page 6, 1994. URL http://arxiv.org/abs/cmp-lg/9406033.

[11] V I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. URL http://sascha.geekheim.de/wp-content/uploads/2006/04/levenshtein.pdf.

[12] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. URL http://link.aip.org/link/SJOCE3/v20/i1/p359/s1&Agg=doi.

[13] S Van Dongen. Performance criteria for graph clustering and markov cluster experiments. *TRINS R0012*, (INS-R0012), 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.9783.

[14] Kris Holt. Study: 13november, 2011. URL http://www.scribbal.com/2011/06/study-13-of-us-internet-users-are-on-twitter-up-from-8-in-november/.