SUPERVISED CLASSIFICATION OF TWITTER POSTS VIA UNSUPERVISED CLUSTERING OF HASHTAGS EECS 545 MACHINE LEARNING FINAL PROJECT PROPOSAL

GREGORY HANDY, DOLAN ANTENUCCI, AKSHAY MODI, MILLER TINKERHESS

Abstract. While Twitter is full of information, retrieving this information and making sense of it is a non-trivial task and has become a focus in machine learning. We develop two tools that are helpful in in unlocking this information: the clustering of hashtags into meaningful topics, and using these clusters to identify the topic of tweets through the use of classification methods naive Bayes, LDA and SVM. In addition to creating a respectable clustering algorithm, a novel similarity measure is developed that is useful beyond the scope of the methods we examined. Our classification step also illustrates... One Line Here about how cool our classification step is

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 [11, 1, 12]. 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 and so 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 [**?**]. We represent each tweet as a vector with one binary dimension for each hashtag in the data set indicating whether or not the tweet contains that hashtag. To simplify the learning problem, we will cluster hashtags into categories and learn over the clusters.

Prediction of a user's future tweet content could be useful in a number of ways. For example, it would allow Twitter to present a customized list of trending topics to each user, allowing them to discover content that is relevant to their interests that they might otherwise not have discovered. Twitter could also present a list of users whose predicted behavior is similar to that of the user, making the assumption that the user might be interested in following those

users. Prediction information could also be used to present relevant advertising to a user, thereby increasing click-through rates over non-targeted advertising.

2. Related Work

The most related work in what we are doing with clustering was done by J. Poschko who focused on clustering of hashtags. He developed the idea that two hashtags are similar if they co-occur in a tweet [**?**]. Based on this idea, he creates a clustered graph with the co-occurrence frequency as the distance measure. We expand upon this idea in our clustering by introducing a novel method for measuring the distance between two hashtags. Additionally, we use a larger set of hashtags and test several clustering methods to instead of focusing on just one.

In other clustering work involving hashtags, Bode et al. use hashtags to cluster users together via multidimensional scaling and hierarchical cluster analysis [2], but only focus on when users are using the same hashtags. Additionally, Carter and Tsagkias use hashtags to build a thesaurus of related terms [3]. SHOULD WE INCLUDE?? I dont think so (Greg)

Other work in clustering of text-related entities typically focus on a bag-of-words model that takes all the words of the entity, along with some dimensionality reduction, to make clustering computationally feasible [8, 5]. This idea has been extended to address the issue of sparsity in tweets by including other words fetched from larger corpuses like Wikipedia, or simply combining several tweets together by a shared attribute such as the tweeter [14, 7, 9, 4]. While these methods to address sparsity can help with clustering of tweets, this does not help with clustering of hashtags.

There has been notable research on the classification of tweets in different contexts. One context attempts to classify the sentiments of different tweets, such as the mood or opinion of the tweet [6]. This requires a considerable amount of work in analyzing the linguistics of the tweet. Our focus of classifying the topic of the tweet, not the sentiment, will not require this analysis, providing an easier tool to classify, while still producing useful information.

Additionally, researchers Mazzia and Juett investigated the recommendation of hashtags based on a tweet's content [10]. They used a modified Naive Bayes classifier to aid in their recommendations. We expand on this idea by recommending hashtag clusters, which allows us to expand our set of supported hashtags.

3. Our Approach

3.1. **Data Description.** Our starting data set is a collection of 476 million tweets from June 2009 to December 2009. This data set was collected by Stanford University through their partnership with Twitter, and then later shared with other research teams. The data contains 17 million unique users.

We begin our data analysis by performing several pre-processing steps to aid in our analysis . We first limit our dataset to English tweets from July and August of 2009, which totals

178 million tweets. From these, we build a list of tweets containing at least one hashtag, which totals around 16 million tweets. Next we remove all non-alphanumeric characters (e.g. commas, periods, etc.) to allow us to simply focus on the *terms* (i.e. words, numbers) and hashtags being used in each tweet. The number of unique hashtags over this time period is 872,402.

The last step of pre-processing was to remove several hashtags that are either auto-generated by external systems, or are *micro-meme* hashtags, which are those used in no discernible pattern. An example of an auto-generated hashtag is #fb which is appended by a particular Facebook application that shares posts on Facebook with Twitter. An example of a micro-meme hashtag would be #followfriday, which is a weekly used hashtag by users to recommend other users of interest.

From this pre-processed data, we are able generate a list of the most frequently used hashtags, along with the co-occurring frequencies of all hashtags and terms with each other.

To simplify calculations during clustering, we limit ourselves to a subset of the top hashtags, along with all the hashtags they co-occur with. This is described in further detail in the Clustering Hashtags section. For our classification step, we limit our focus to a subset of tweets.

3.2. **Clustering Hashtags.**

3.2.1. *Difficulties of Clustering.* Clustering is inherently a hard task, and clustering hashtags is no exception to this rule. There are a number of variables that make clustering hashtags very difficult. The first major difficulty is the fact that they are not required to be defined words. For example, the hashtag #p2 is a popular marker if a tweet contains politically progressive thoughts. In most cases, hashtags are a concatenation of words such as #iamthemob. There exist measures such as the Wu-Palmer distance and path distance similarity that are distances that are based on the synonyms of the words being examined. Unfortunately, this feature eliminates the possibility of using this measure as central component in a clustering algorithm, such as K-means. Our algorithm will still use this concept, but only in a very limited scope.

The fact that hashtags are usually concatenations also makes it difficult to apply another popular metric to compare words, known as Levenshtein distance (i.e. edit distance). As an example, this distance would say that the hashtags #iamblessed and #iamthemob are close to one another, when in reality, they are very different. We actually implemented a method centered on this distance, and the poor results backed this observation.

Further, the set of hashtags is constantly increasing in size, with new, "trendy" hashtags appear everyday. This fact forces us to create an algorithm that captures the essence of most hashtags used, while remain tractable. To account for this, we decided to focus our attention on the 2,000 most used hashtags, and clustered only over this set. This decision was a directly consequent of the final clustering algorithm, and will result in the clustering of more than 2,000 hashtags.

3.2.2. *Co-Occurrence Relation.* The paper written by J. Poschko developed the idea that two hashtags are similar if they co-occur in a tweet [**?**]. Intuitively, this concept makes sense. Two hashtags are inherently similar if they are contained in tweets that discuss the same topic, and this fact could not be any stronger than if they appear in the same exact tweet. To back this intuition, the paper 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. It was mentioned previously that the Wu-Palmer distance is a poor distance for hashtags as a whole, since there would be many unknown distances between hashtags that are not words. However, it does provide a good measure if you only consider the set of hashtags that is made up of well-defined words. From this fact, we make the assumption that if 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-occurring lists for the 2,000 most used hashtags, we used the natural language toolkit (NLTK) in python to find the Wu-Palmer distance for each hashtag in each list. We then took the average over all of the lists. The final value is found in Table 1, along with the baselines found by Poschko. This table verifies that the claim by Poschko applies

TABLE 1

	Average S_{WP}
Co-Occurrences	.40
Baseline (Random Words)	.16

to our data set. Clearly, the hashtags found in co-occurrence list are much more similar than random words.

3.2.3. *Minimizing the Level of Noise.* However, even though this applies to the list as a whole, it does not mean that there is not still a considerable about of noise present. For example, #photography is a popular hashtag one uses to denote a recently posted picture, and another hashtag is sometimes used to describe the place the picture is taken, such as #iran. However, as this example illustrates, the fact that these hashtags co-occur does not mean that these two hashtags are similar by our definition of similarity. To help minimize the effect this has on later stages, let n_{ij} be the total number of co-occurrences hashtag i has with hashtags j. Hashtags A and B are kept on the co-occurring list if and only if

$$min(\frac{n_{AB}}{\sum n_{Aj}}, \frac{n_{BA}}{\sum n_{Bj}}) > .05,$$

otherwise it is removed from the list.

Even after this step, it is still possible that many noisy relations exist; therefore we do another level of filtering, by comparing the contents of the respective co-occurring lists. Let m be the total number of hashtags that occur in both hashtag A and hashtag B 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

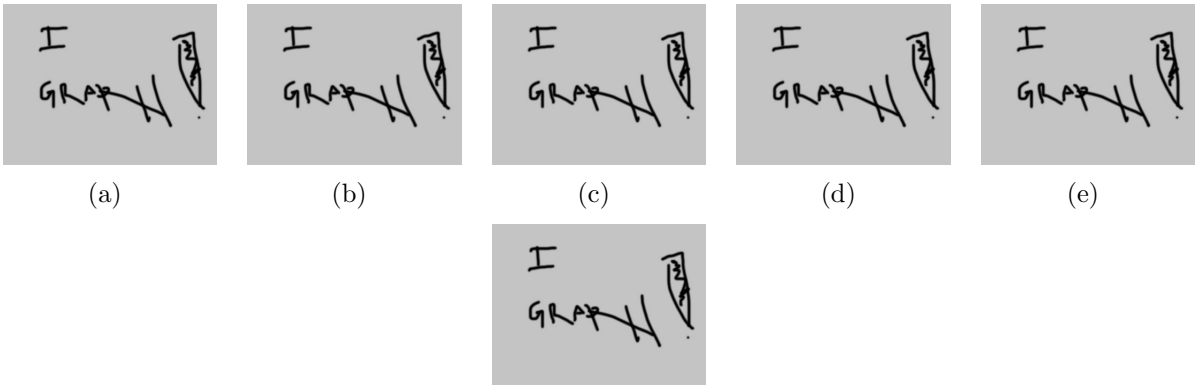
$$min(\frac{m_A}{\sum n_{Aj}}, \frac{m_B}{\sum n_{Bj}}) > .2.$$

Note that by the first level of filtering, this minimum is at least .05. Comparing each list takes a considerable amount of time to run, and this bottleneck forced us to constrain our focus to only 2,000 hashtags. However, it is easily parallelizable, and when program efficiently, would allow a much greater number of hashtags to be considered.

3.2.4. *Defining the Similarity Measure.* Using the Wu-Palmer distance and filtering methods described, we are confident the lists of co-occurring hashtags that remain imply some level of similarity between hashtags in these lists. As a result, we define the following similarity measure between hashtags A and hashtags B:

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

This function fits the axioms of a similarity measure. S(A,B) = S(B,A), and S(A,B) $\geq$ 0, and based on filtering process it will actually be greater than .05. 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.



(a) (b) (c) (d) (e)

(f)

FIGURE 1. Subfigure (a) represents the original undirected graph proposed by the co-occurring lists. Subfigures (b) and (c) consists of only 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).

3.2.5. *Spectral Clustering and METIS.* With our well defined similarity measure, and thus 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 a graph program named METIS that claimed to perform 10% to 50% better than spectral algorithms [**?**]. This algorithm is based on 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.

3.3. **Classification.** In the classification step, we will attempt to classify various 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 dictionary of all unique words (removing common stop words 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 of the i^{th} word in the tweet.

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 a software package for Python named *scikit-learn* to reduce the dimensions in our data. For our dataset containing 100,000 tweets, we had approximately 46,000 dimensions. We reduced this to 100 dimensions using PCA. Once we had reduced the number of dimensions, it was possible to apply some standard classification algorithms to the data. We applied the following algorithms to the reduced-dimension data.

3.3.2.1. Naive Bayes. Applying the Naive Bayes algorithm, we were getting very poor performance. This is because PCA creates real-valued feature vectors whose values don't always mean much. Hence we binarized the data using a simple strategy: all positive values in the feature vector took the value 1, and all negative values took the value 0. Using this new data we were able to get reasonable performance using the Naive Bayes classification algorithm.

3.3.2.2. Linear Discriminant Analysis. When using LDA, we were able to get acceptable performance using just the reduced-dimension data. We didn't need to create a binary representation of the data.

3.3.3. *Classification using SVM algorithm optimized for sparse vectors.* Another method we used to circumvent the high dimensionality of the feature vectors was using 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 a Radial Basis Function as the kernel for the SVM. Since SVM is an inherently a binary classifier, we use a one-versus-all strategy to make it a multi-class classifier. This was also implemented using the software package *scikit-learn*.

3.3.4. *Majority Vote classification.* As a baseline measure, we also did classification by majority vote. This method simply involved looking at all the training data, and predicting each test point that class which occured most in the training data.

4. Experiments and Results

4.1. **Clustering.** As mentioned previously, it decided that our focus would be placed on clustering the top 2,000 most used hashtags. Decreasing this number eliminated the breadth that our clusters would cover; however increasing this number made the preprocessing step intractable. Several trails were performed on a set of 5,000 top used hashtags, without running the preprocessing step, but most of the hashtags grouped into 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 in the preprocessing described.

Upon completion of the preprocessing step, only 1,557 hashtags remained. This provided our clustering algorithms with a 1,557 x 1,557 similarity matrix. Unfortunately, there is no real intuitive way to pick the number of clusters. 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. As result, we tested several different values for K, and evaluated the size of the clusters that were formed. A small value for K would result in hashtag clusters that were not indicative of one topic. Picking a large value of K would separate clusters that should actually be in the same group. It was concluded that K=300 provided the best combination of these two features.

One method of cluster evaluation was to pick out randomly 100 out of the 300 clusters, and evaluate them 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 the 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 --- 5 (collection of all football teams)

amazing comic comics cool funny humor strange voss webcomic webcomics weird
```

```
--- 3 (clearly comics are represented, but there are random words such as
amazing and weird)

bangladesh car classifieds fatpeoplearesexier mobile motorcycle motorsport
nokia shopping used --- 1 (list looks very random)
```

Table 2. 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

Its clear from the table that spectral clusters performs the best, with normalized spectral close behind. For normalized spectral, its interesting to note that the clusters were found around two the extremes, with not many scoring the middle values. METIS had the additional issue that it contained many clusters that should have been clustered together. For example, it contained 5 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 the big topics readily. Its also possible the METIS would avoid this issue with K was chosen to be a smaller number.

4.2. **Classification.**

4.3. **Discussion.**

5. Future Work

5.1. **Clustering.** Although we have shown that our clustering method has promise, there are a number research points that can be explored to improve the method. During part of the preprocessing step, as 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 possible that this algorithm would be less effected by noise, which would eliminate the preprocessing step and allow a large number of hashtags to be clustered. Parallelizing the preprocessing step would also allow for this possibility.

A better way of choosing the number of clusters should also be examined. Further, increasing K was meant to increase the number of topics the clusters represented. However, instead of splitting large clusters apart into smaller topics, it would simply pull one or two hashtags out at a time. This was most likely do to the fact that K-means was always done on the full graph. It would potentially be better to first apply K-means to the whole set, and have K equal a small number. Then for each of these clusters, perform a localized K-means. This method should avoid the issue of creating clusters that are too small in size.

5.2. **Classification/Prediction/What else can we do with this?** Future work with classifying and prediction using our cluster data focuses on three areas. First,

6. Conclusion

In conclusion, we rock.

References

[1] Alexa. Alexa top 500 sites on the web, 2011.

[2] L. Bode, A. Hanna, B. Sayre, J. Yang, and D. Shah. Mapping the Political Twitterverse: Finding Connections Between Political Elites. In *Networks*, pages 1–25. Southern Illinois University Carbondale Year, University of Wisconsin-Madison, 2011.

[3] S. Carter and M. Tsagkias. Twitter hashtags: Joint Translation and Clustering. *Human Factors*, pages 1–3, 2011.

[4] Q. Chen, T. Shipper, and L. Khan. Tweets mining using WIKIPEDIA and impurity cluster measurement. *Information Systems Journal*, pages 141–143, 2010.

[5] M. Cheong and V. Lee. *A Study on Detecting Patterns in Twitter Intra-topic User and Message Clustering*. IEEE, Aug. 2010.

[6] D. Davidov, O. Tsur, and A. Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. *staffscienceuvanl*, (August):241–249, 2010.

[7] J. Hu, L. Fang, Y. Cao, H.-J. Zeng, H. Li, Q. Yang, and Z. Chen. Enhancing text clustering by leveraging Wikipedia semantics. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08*, page 179, New York, New York, USA, July 2008. ACM Press.

[8] A. Karandikar. Clustering short status messages : A topic model based approach. *Work*, 2010.

[9] N. N. Liu, C. W. Bay, H. Kong, and K. Zhao. *Transferring Topical Knowledge from Auxiliary Long Texts for Short Text Clustering*. ACM Press, 2011.

[10] A. Mazzia and J. Juett. Suggesting hashtags on twitter. EECS 545 Project, Spring 2011, 2011.

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

[12] Twitter.com. Your World, More Connected, 2011.

[13] S. Van Dongen. Performance criteria for graph clustering and markov cluster experiments. *TRINS R0012*, (INS-R0012), 2000.

[14] W. X. Zhao, J. Jiang, J. Weng, J. He, and E.-P. Lim. Comparing Twitter and Traditional Media using Topic Models. *New York*, 6611:338–349, 2011.