Twitter Tweaker – Twitter-based Recommendation System

Shruti Sahu

Computer Science

Iowa State University

shruti@iastate.edu

Souparni Agnihotri

Computer Engineering

Iowa State University

souparni@iastate.edu

Tanmay Gore

Computer Science

Iowa State University

tgore03@iastate.edu

**Abstract**

A recommendation system for twitter that identifies new followers for a user based on the popularity of words within his tweets. The described method maps the tweets to TF-IDF vectors in a sparse zero matrix and clusters this data using k-means clustering. The method then computes a new TF-IDF vector array for the new tweet and classifies it to one of the clusters obtained from k-means.

**Keywords:** TF-IDF, K-Means, Twitter, Recommendation.

1. **Introduction**

Twitter is a widely used social media platform that has become one of the primary sources of information for news, entertainment, and other media. Users follow other twitter profiles to gather this information. However, it can be challenging for inexperienced users to get started with Twitter as they may not know whom to follow to find articles that interest them. In addition, existing users might miss some important updates as they may not be following the right users. Thus, we find a need for a recommendation system that would automatically notify users about other users he/she might be interested in following.

This project includes a novel method that recommends new users to follow based on the current user’s interests. The interests of the user are obtained from his/her most recent tweets.

This project explores two main ideas – TF-IDF and k-means clustering. TF-IDF stands for *term frequency-inverse document frequency*, and the TF-IDF weight is often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. (http://www.tfidf.com/ n.d.) The importance of a word is proportional to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

The following are the equations for TF and IDF respectively:

… (1)

… (2)

The TF-IDF is the product of equation (1) and (2).

The second concept that we explored is k-means clustering. (https://www.datascience.com/blog/k-means-clustering n.d.) It is an unsupervised learning technique (i.e. no labels for the data are provided) that is used for clustering similar data into groups. The number of groups made depend on the value k that we choose.

Finally, we used Cosine and Jaccard Similarities, which are some of the most standard similarity measures across a set of documents. This was implemented as a means of testing against k-means clustering.

1. Methodologies

To classify a new tweet, we perform three major steps: (1) Extract 1000 tweets from twitter API (2) Computing TF\_IDF to group these sample tweets into k groups using k-means clustering and (2) Classify the new tweet based on the Euclidean distance between the new tweet and the existing data set.

* 1. Extraction of the sample tweets

Twitter provides two APIs for retrieving tweets and user information. The Streaming API provides methods to retrieve tweets in bulks and filter them out based on location, keywords, users, etc. We used the Streaming API for collecting 1000 sample tweets in English and store it in a file in JSON format. The REST API provides methods to retrieve user-specific information such as user’s latest tweet, contact information etc. We use this API to get user’s most recent tweet to classify them into the created clusters.

* 1. Grouping the sample set into k clusters

For clustering twitter data, we had to first assign a TF-IDF score for every word in a tweet. This would enable us to have a vector representation of every tweet that was obtained. Once the TF-IDF scores were obtained, a set containing every word was made. Let’s call this set “*word\_set*”. Every word in *word\_set* was assigned a unique index (*index\_word*). Next, a sparse zeros array containing dimensions [number of tweets, length of *wordSet*] was generated. We iterated through the TF-IDF scores for every word in a tweet and added them to the specific index as determined by *index\_word*. This was done for every tweet we gathered, and we passed the resultant matrix into the K-means algorithm.

A screenshot of a cell phone

Description generated with very high confidence

Figure 1: Block diagram illustrating clustering phase

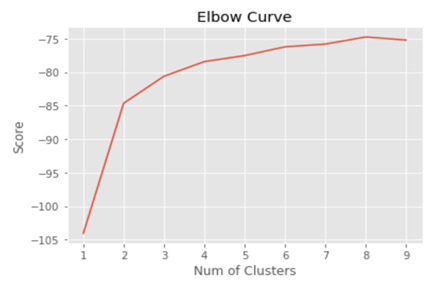
In our implementation of the k-means algorithm, we decided to set the value of k equal to six. This conclusion was reached after analyzing the percentage of variance in the clusters as the value of k increases.

Figure 2: Number of clusters vs variance

(Score)

The graph in figure (1) illustrates the percentage of variance in the clusters (score) as the number of clusters increases. When k=1, we see that the percentage of variance is very high (105%) but it drops drastically when we have 2 clusters. We observe that somewhere between k=6 and k=7 onwards, the percentage of variance does not change too much. Hence, we concluded that 6 would be the optimum number of clusters we need to minimize inter-cluster variation and get the best clustering results.

* 1. Classification Methodology

This phase of the algorithm deals with classifying the new tweet to the clusters obtained from the previous phase. When a user posts a new tweet, the tweet is obtained from the Twitter API. We first filter out the hashtags, URL’s, and stop words from the text of this tweet. Next, we generate the TF-IDF value for each word obtained in the tweet.

To generate TF-IDF values, we acquire the TF-IDF values and list of unique words generated in from our first iteration of computing the vectors, and k-means clustering.

Using the TF-IDF values generated for the new tweet, we populate the sparse zero array with these values using the same methodology described in section 2.2. The sparse array of zeroes specific to the new tweet is computed and this will have dimensions [1, length of *wordSet*]. We use this matrix to assist us in classifying the new tweet into a cluster. This is accomplished by computing the Euclidean distance between the centroids of all clusters to each data point in our matrix. The new tweet is assigned the cluster in which the Euclidean distance between the centroid and the cluster is minimized. Once the label is known, we recommend users whose tweets also belong to this cluster. This process is then repeated for different tweets.

1. Experiments

In this section, we collect different sets of 1000 tweets from the Twitter API to test our model. We compare the results of our model with Jaccard and cosine similarities to evaluate the accuracy of our k-means clustering and classification technique.

* 1. Jaccard Similarity

The basis of Jaccard similarity lies in finding the ratio between thenumber of words that are the same across two tweets and the total number of unique words across the two tweets.

Once these similarities are computed, we compare the labels of all tweets and the label that has the maximum number of tweets assigned to it will be the label to which the new tweet should be grouped into. After running the classification algorithm on the new tweet, we obtain the label for that tweet. In our experiments, we observe that the classification algorithm classifies the new tweets to the cluster that contains the most number of similar tweets as computed by Jaccard similarity. As a result, we are using Jaccard similarity as a basis for testing the performance of our k-means clustering.

* 1. Cosine Similarity

Cosine Similarity is a method of measuring similarity between two documents. The following is the formula for computing between two documents and :

… (1)

where and are the binary vector of the documents i.e. the length of each vector is equal to the size of the union of both the documents. All the words in the union set are indexed and if the word appears then the binary vector has 1 in that position else has 0. Also, represents the L2 norm of the vector, i.e. the length of the vector.

In our recommendation system, we use cosine similarity to measure the similarity between the new tweet and the sample tweets (i.e. our training dataset). All the similarity measure between each sample point and the new tweet is then compared and we find the tweet in the sample set that has the highest similarity with the new tweet. The label of this tweet is compared with the label that the new tweet gets from our classification model. In this way, we evaluate our classification algorithm.

1. Results & Discussions

After extensive testing and analysis, we found some interesting results from our clustering and similarity measures. The k-means performed well only when the words in the new tweet existed in the word set. Otherwise, when an unfamiliar word is introduced, the TF-IDF value for that word would not be added and there was a higher chance that tweet would get grouped into the wrong cluster.

However, when the new tweet contains familiar words, our K-means seemed to perform very well and would cluster it with very similar tweets. This was also further verified by Jaccard similarity which showed us that some of the results were accurate, while others were being grouped into the second or third best cluster. An interesting finding was that the label which the k-means assigned to the new tweet was the label to which Jaccard similarity had the maximum number of similar tweets classified to. The below output shows this result:

We can see that cluster 3 has the most number of common words and as a result, our k-means algorithm would group the new tweet into cluster 3.

This way, we were able to cross verify the results and test the working of our k-means clustering.

Using the information from the group the new tweet was clustered into, we were not only able to get other similar tweets but were also able to find the users corresponding to those similar tweets and recommend following those users.

|  |  |  |
| --- | --- | --- |
| Cluster No | Count of common words with new tweet | Jaccard Similarity with each cluster |
| 0 | 1 | 0.1 |
| 1 | 0 | 0 |
| 2 | 4 | 0.0728 |
| 3 | 39 | 0.0506 |
| 4 | 4 | 0.0708 |
| 5 | 0 | 0 |

Table 1: Jaccard Similarity Results for new tweet clustered in cluster 3

1. **Conclusion**

This project focused primarily on the k-means clustering algorithm and grouped similar tweets together. The novelty came in the fact that we were using the user’s most recent tweet and clustered it into groups such that we would be able to recommend the user other people to follow based on the content of his/her tweet. In addition, we also explored other similarity measures like cosine and Jaccard similarities and used them to verify the results of our k-means clustering. We found that the clustering was giving us at least second or third best results when tested with Jaccard. However, when we passed in the same tweet multiple times we observed that it was always successfully classified in the same cluster, hence proving the credibility of our system.

In the end, we were able to develop an efficient recommender system that would correctly group a user’s most recent tweet into a cluster containing similar tweets and we were able to find the users corresponding to those similar tweets and recommend to the user who he/she should follow.

1. **References**

http://www.tfidf.com/ ,20 Nov 2017

https://www.datascience.com/blog/k-means-clustering , 22 Nov 2017