Social Media Clustering

Spencer Hanson

Zach Brueske

Auston Schendzielos

**Abstract**

Social Media outlets are quickly becoming some of the biggest online entities. Mediums such as Facebook and Twitter have become key sources for news and trends in regards to popular culture. Through techniques related to data mining (i.e. classification, clustering), these trends can be easily found and then news sites that report on pop-culture can write various types of articles on them. In our case, we chose to use three clustering algorithms on a set of data taken from Twitter. The first algorithm chosen was PAM (partition around medoids), which is a k-cluster based algorithm. Next, we chose hierarchical group average, an agglomerative clustering technique that uses a proximity matrix. The last algorithm, OPTICS, is a density-based algorithm that builds upon another density-based algorithm, DBSCAN. In this paper, we will explain in more detail how the algorithms work and the results that they produced, with a final conclusion on which algorithm was best suited for this type of data set.

**Introduction**

Our data set was taken from Twitter after an observation period of seven days. Because Twitter is such a popular social media outlet, this makes the data set extremely large; 140,707 instances of words that were followed during the observation period. Each word is listed as either a buzz or non-buzz word and our goal was to create pure clusters with the words. In the Data section of this paper, we will go over more thoroughly our data set and the preprocessing techniques used.

Partitioning around medoids (PAM) was an algorithm developed by Leonard Kaufman and Peter Rousseeuw in 1990 (Ng & Han). PAM’s approach is to determine a representative object for each cluster, called a medoid, which is centrally located within each cluster. This centrally located object is meant to be less susceptible to noise and outliers than a centroid from the basic k-means clustering algorithm. The biggest problem with PAM is that is very poor when ran on a large data set, which will go into more detail about in the following sections.

Hierarchical clustering aims to group data over various scales by creating a dendrogram, which is essentially a cluster tree. The tree is a multilevel hierarchy of clusters, where each level is comprised of clusters that will form together into one cluster on the next level. With the tree, it’s possible to select the desired amount of clusters appropriated for a certain application.

OPTICS (Ordering Points to Identify the Clustering Structure) was founded at the University of Munich. OPTICS is a density based clustering algorithm that takes into account

//austonlol

**Data**

The data set chosen for our research was taken from Twitter. There are 140,707 instances of words that were observed over a period of seven days. Each word has 11 attributes that were observed every day, which is 77 total attributes for each instance. Additionally each word is either a buzz or non-buzz word. For a word to be considered “buzzing,” it has to have been interacted with at least 500 times during the seven-day observation period. In our data set, the buzz to non-buzz ratio is 20/80 in terms of percentage. This means we have a skewed data set because 80% of the words are not buzz words.

The 11 attributes that each word has are as follows:

**Number of Created Discussions**

Measures the number of discussions created during the observation period

**Author Increase**

Measures the number of new authors interacting on the instance’s topic

**Attention Level**

Measure of attention paid to the instance’s topic on social media (measured by author)

**Burstiness Level**

The burstiness level for a topic

**Number of Atomic Containers (number of “tweets”)**

The total number of atomic containers generated through the whole social media on the instance’s topic

**Attention Level**

Measure of attention paid to the instance’s topic on social media (measured by contributions)

**Contribution Sparseness**

Measure of the spreading contributions over discussion for the instance’s topic

**Author Interaction**

Average number of authors interacting on the instance’s topic within a discussion

**Number of Authors**

Measures the number of authors interacting on the instance’s topic

**Average Discussion Length**

Direct measure of the average length of a discussion belonging to the instance’s topic

**Average Discussion Length**

Measure of the number of discussions involving the instance’s topic

**Pre-processing**

Because the data set is so large and each instance has 77 attributes, the first we thing we did to reduce the size was take the median value of each of the 11 attributes. This brought the total number of attributes for each word down from 77 to 11. Next, we created smaller data files of randomly selected instances taken from the main data set. This was done because no matter how efficient or inefficient each algorithm was, they were taking too long to complete a run on a data set that is 140,707 instances big. The smaller files we created maintained the same buzz to non-buzz ratio as the main data set and our algorithms could complete various runs on them. We created a data set of 1,000 instances, one of 2,000, 4,000 and finally, 8,000.

**Algorithms**

In this section, we will go into much greater detail of the explanation of each algorithm that we used, along with strengths and weaknesses of each in regards to our particular project.

**PAM**

PAM is a k-means algorithm that uses centrally located data objects to represent each cluster, known as *medoids*. The medoids are initially selected randomly and then swapped with every other point in the data set until it finds the most centrally located object for each cluster. Here is a brief summary of the algorithm:

**Use real object to represent the cluster**

**Select *k* representative objects arbitrarily**

**For each pair of non-selected object *h* and selected object *i*, calculate the total swapping cost *TCih***

**For each pair of *i* and *h*,**

**If *TCih* < 0, *i* is replaced by *h***

**Then assign each non-selected object to the most similar representative object**

**repeat steps 2-3 until there is no change**

To find the k medoids, PAM begins with a random selection of objects. Then in each step, a swap between a selected object Oi and a non-selected object Oh is made, as long as the swap would result in an improvement of the quality of the clustering. To calculate the effect of such a swap, PAM computes cost Cjih for all non-selected objects Oj. Depending on which of the following cases Oj belongs to, Cjih is defined by one of the equations below.

First case:

Oj belongs to the cluster represented by Oi and Oj is more similar to the next closest medoid, Oj2 than Oh. Thus, if Oi is replaced by Oh as a medoid, Oj would belong to the cluster represented by Oj2. So the cost of the swap as far as Oj is concerned is:

Cjih = d(Oj, Oj2) – d(Oj, Oi)

This cost will always be a positive number.

Second case:

Oj belongs to the current cluster represented by Oi but it’s more similar to the potential new medoid instead of its next closest one. Then, if Oi is replaced by Oh, Oj would belong to the cluster represented by Oh.Thus, the cost for Oj is given by:

Cjih = d(Oj, Oh) – d(Oj, Oi)

This number can be positive or negative depending on whether Oj is more similar to Oi or Oh.

Third case:

Oj belongs to a different cluster and it’s more similar to its own medoid than Oh. Then if Oi  is replaced by Oh, Oj would stay in its own cluster. Thus the cost would be:

Cjih = 0

Fourth case:

O­j belongs to another cluster but it is more similar to Oh than its own medoid. Then replacing Oi with Oh would cause Oj to jump to the cluster represented by Oh. Thus, the cost is:

Cjih = d(Oj, Oh) – d(Oj, Oj2)

This number will always be negative.

Combining the four cases above, the total cost of replacing Oi with Oh is given by:



PAM works very well on small data sets but it is not efficient in dealing with medium and large data sets. The time complexity for PAM is O(k(n-k)2) for only one iteration. This is one of the main reasons we reduced our data set down to sizes such as 1,000 and 4,000. PAM would take far too long to run on a data set the size of 140,707 instances.

**Group Average**

Group Average is a type of Hierarchical clustering that using the average distance from all the points in a cluster to all the points in another cluster to find the shortest distances between clusters. It was ran as an agglomerative algorithm, or “bottom up” approach. This means each of the data points started as their own cluster and were merged together until the desired number of clusters was reached. Here is a summary of the algorithm:

**Compute the proximity matrix**

**Let each data point be a cluster**

**Repeat**

**Merge the two closest clusters**

**Update the proximity matrix**

**Until only the desired amount of clusters remain**

Group Average computes the initial proximity matrix of size n by n to see which points will be the first to merge. It then takes the smallest distance and adds those two points together in a cluster. To update the proximity matrix it removes the two points that were just merged from the matrix and adds the new cluster to the end of the matrix after reducing the size by 1. This way it will not compute distances it has already done before. After the proximity matrix is updated it will again find the shortest distance and merge those two points or clusters. When merging the clusters Group Average has to go through each point in the cluster and find the average distance between all points in both clusters. It uses the following formula to compute distances between clusters:

 \frac{1}{|A| |B|} \sum_{a \in A }\sum_{ b \in B} d(a,b). 

Group Average is a simple clustering algorithm that is a compromise between single and complete link. It also has advantages over using Min and Max because it is not susceptible to noise and outliers, and can also handle globular clusters really well.

However, the fact it handles globular clusters well is also a negative. This is because it is really biased towards those types of clusters. The other bad aspect about Group Average is that it does not work well when using a large data set. This was not good for our project because the data set we were initially using had 140,707 instances. Even when we reduced it down it would only run well on 1,000 or 2,000 instances, and would take too long to run on sizes over 2,000.