Automatic Clustering of Celebrities using Tweets

Kenny Ning
kning1@swarthmore.edu

Jon Cronin
jcronin1@swarthmore.edu

Abstract

This paper attempts to automatically classify celebrities into similar groups based on tweets that refer to that celebrity. We created a list of 100 celebrities that was equally distributed across four different professions: athletes, musicians, politicians, and actors. Tweet data for each of these celebrities was collected from November 27 to December 6, and we ended with about 1.81 million tweets total. Next, we examined each person's tweets and recorded how frequently certain unigrams in those tweets were present. Using tf/idf weight vectors for each celebrity, we calculated the distance (cosine similarity) between celebrities and performed different clustering techniques to see which celebrities were most alike. Overall, we found that people tweet about different kinds of public figures differently, and that looking at unigram frequencies in twitter data is useful to determine in which category a celebrity belongs.

1 Introduction

Twitter is a very large community of users posting content about daily events in their lives and around the world. Recently, celebrities have become a major presence on Twitter, and millions of people follow and tweet about major public figures. People tweet in large volumes about athletes' games, actors' movies, musicians' songs and albums, and politicians' elections and bills every minute of every day. In this paper, our goal is to see if people talk about similar celebrities using similar words and hashtags, and then to see if we can group similar celebrities together based solely on these tweets.

We use a keyword search of a celebrity's name to gather tweets that we think will have an overall trend related to that celebrity's profession, and hypothesize that the tweets between two celebrities who are closely related (two basketball players) will look more similar than the tweets for celebrities who do not have much in common (a basketball player and a governor). Go et al propose a similar method to perform a different classification task: sentiment analysis. They gather tweets using emoticons, and propose that tweets with 'happy' emoticons (":)", ":D") will have positive sentiment, and that tweets with 'sad' emoticons (":(", ":'(") will have negative sentiments (2). They seperate tweets into two categories using certain keywords; similarly, we seperate tweets into four categories using names of celebrities related to those four categories.

We calculate which celebrities are in which groups using clustering algorithms, a project which resembles Muntean et al. They clustered tweets based around hash tags to try to find which hash tags were most similar (3). Hash tags are tags that users put onto tweets to identify them with larger movements or trends. For example, the tag "#election2012" identifies the tweet is about the 2012 election. Other hash tags are used about more general topics, such as #music or #news. While Muntean et al. tries to find similar hash tags by looking at

tweets that contained those tags, we try to find similar celebrities by looking at tweets that contained that celebrity's name.

Choudhury et al have a project that uses similar techniques described above to perform a different kind of classification. They attempt to perform an automatic classification of Twitter users into three categories: organizations, journalists/media bloggers, and ordinary individuals (1). The classification method that they used was *k* Nearest Neighbors, where each user is represented as a vector of features. Overall, they were able to classify users according to their definitions fairly accurately, and were able to use these definitions to obtain meaningful analyses of user behavior. For example, they saw that ordinary individuals tend to use more personal and sentiment-based language and interact with other users more often.

Overall, the idea in these papers, and our project as well, is that people use language to signal broader ideas, such as a sentiment or a topic. Our hypothesis is that people will use similar language and hash tags to identify similar kinds of topics. Ideally, the general Twitter "buzz" surrounding a particular person's name should provide clues as to what type of celebrity he/she is, although it is possible that it will signal different groupings than our proposed ones, which are slightly arbitrary.

2 Methods

2.1 Tweet Data Collection

First, a list of 100 celebrities was compiled, equally distributed across four different professions: Sports, Music, Politics, and Movies. In other words, the list of celebrities includes 25 athletes (e.g. Lebron James, Derek Jeter), 25 musicians (e.g. Lady Gaga, Justin Bieber), 25 politicians (e.g. Barack Obama, John Boehner), and 25 actors/actresses (e.g. Leonardo DiCaprio, Natalie Portman). The choice of these celebrities was created somewhat subjectively, but was influenced by followers/talked about statistics on twittercounter.com.

After compiling this list of celebrities, we used a Python web mining module called Pattern. Utilizing this module, we wrote a python script (adapted from a version that Sam Clarke '13 had written) that collected approximately 100 tweets for each of the 100 celebrities every 40 minutes. Note that these results are tweets that contain the celebrity's name in it, not particular messages tweeted by the celebrity. Our script handles for cases of duplicate tweets, but it does not handle retweets. Each of these tweets are stored into a Datasheet structure (provided by the Pattern module) and written to a file. We ran this script over the course of a week and a half (November 27, 2012 - December 6, 2012), ending with approximately 1.81 million tweets.

Category	Celebrity	# of Tweets
Sports	Nick Swisher	7817
Sports	Tiger Woods	18446
Sports	Kobe Bryant	22020
Music	One Direction	26783
Music	Justin Bieber	27838
Music	Jay-Z	29264
Politics	Jerry Brown	6012
Politics	Elizabeth Warren	9932
Politics	Barack Obama	24585
Movies	Zach Braff	4139
Movies	Jackie Chan	19882
Movies	Will Smith	26423

Table 1: A sample of some of the celebrities in our dataset and the # of tweet searches returned. The distribution of tweets is skewed, especially across different categories.

With 1.81 million tweets totaling across 100 celebrities, we would expect around 18,000 tweets for each celebrity. However, the distribution of these tweets across all the celebrities is fairly skewed. For example, based purely on popularity and cultural relevance, a search for tweets about "Jay-Z," an international figure, will most likely return more results than searches for "Jerry Brown," a California politician. Table 1 provides a sample of our dataset.

Out of all categories, we see that musicians tend to have the most tweets, on average, and politicans tend to have the least tweets, on average. This skewness provides some information on what types of tweets are more common. However, this may also affect our clustering results, considering that we will have some celebrities that simply won't have as much informative data to cluster on (e.g. Zach Braff).

2.2 Building tf/idf vectors

Once we have tweets saved for every person, we can begin to create our tf/idf vectors. First, we have to get a list, for every person, of the unigrams that appear in all of their tweets, along with a frequency for each unigram. These unigrams are cleaned in two ways: punctuation is removed (although not hashtags), and all words are lowercase (see section 6 for other possible ways to clean tweets). We create this list as a dictionary, with keys of unigrams and values of frequencies.

Once we have this list for every person, we have to calculate the tf/idf value for each of the unigrams. The term frequency (tf) is saved in each person's unigram dictionary. The document frequencyis computed by looking at every unigram that appears across all people's tweets. We add up, for each unigram, the number of people that have a frequency of one or greater for that unigram. For example, a unigram like "the" will have a document frequency of 100, as it should appear for every person. A unigram like "#nba," on the other hand, will likely appear only in the NBA players' tweets, which would yield a document frequency of around 9 (the number of NBA players).

Once we have the document frequency, the inverse document frequency is just the log of the number of documents (100) divided by the document frequency. We then multiply the idf by the term frequency to get the tf/idf value. This value is saved in each person's vector, which in our implementation is a dictionary with keys of unigrams and values of tf/idf weights.

3 Experiments

Once we have a tf/idf vector for each person, we can use clustering algorithms to group celebrities based on the similarity of these vectors. For this project, we wrote an algorithm that computed k-means clustering, and also used Python's hcluster module to get a hierarchical clustering and another flat clustering.

3.1 K-means Clustering

K-means clustering is an algorithm that uses expectation-maximization techniques to create k clusters of documents. For our purposes, as we divided the celebrities into 4 groups initially, we chose a k of 4, to see if they could be clustered back into their original groups, and also k values of 6 and 8 to see if these gave us better results. To choose these initial seeds, we tried two different techniques. The first was to pick them randomly from our 100 celebrities, and the second was to pick one from each group (also done randomly within each group).

Once we had the initial seeds, we computed the distance from these centroids to every other celebrity. We used cosine similarity to compute the distance between two vectors. We then assigned each celebrtiy to the centroid closest to them to create k different groups of celebrities. Next, we recalculated the centroids by taking the averages of every vector in each group. Once we had new centroids, we repeated the above process until group membership didn't change (i.e. the new centroids were the same as the old centroids).

3.2 Hierarchical Clustering - hcluster

The Python module hcluster has many methods for computing clusterings of data. We adapted a function written by Ameet Soni to create a hierarchical clustering of our celebrities. The function takes in a matrix where each cell in the matrix holds the distance (cosine similarity) between the vector representing the column and the vector representing the row. It also takes a list of labels which get matched up with the column/row indices. This matrix is then turned into a dendrogram which shows how close different celebrities are to each other. In general, celebrities which are next to each other are more similar than ones who are far apart.

3.3 Flat Clustering - hcluster

Hcluster also has a method for computing a flat clustering of a dataset, the output of which is comparable to our k-means clusterings. To compute the flat clustering, hcluster takes as input the hierarchical clustering and then assigns celebrities to groups based on the cophentic distance, a measure of the distance between different branches of the dendrogram, of celebrities within each group.

3.4 Evaluation Techniques

To evaluate how good our clusters are, we use a variety of measures: Purity, precision, recall, and f-value. We define these terms below.

Method	k	Purity	Precision	Recall	f
k-means, random	4	.583	.487	.624	.547
k-means, seeds from each group	4	.748	.628	.714	.668
k-means, seeds from each group*	4	.890	.812	.822	.817
k-means, random	6	.688	.552	.566	.559
k-means, random	8	.722	.606	.485	.539
hcluster - flat	4	.700	.595	.827	.692
hcluster - flat	6	.820	.709	.725	.717
hcluster - flat	8	.830	.703	.664	.683

Table 2: The star (*) indicates that the authors chose these seeds specifically (not randomly). These seeds were Kevin Durant, Mariah Carey, Newt Gingrich, and Leonardo DiCaprio.

Purity is a measure of how "pure" each cluster is, or how not contaminated by wrong celebrities a group is. Purity is calculated as follows: for each cluster, we find the highest represented category and assign the whole cluster to that category. Then we sum up all of the instances a person was sorted in the right category and divide that by the total number of people. For example, if we have 2 clusters, the first of which has 9 athletes and 1 musician, and the second of which has 8 politicans and 2 actors, then our purity would be $\frac{9+8}{20} = \frac{17}{20}$.

To calculate Precision and Recall, we need to define what constitutes True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) in this data set. For each unique pair in our list of people (which is ${}_{100}C_2$ pairs for our list of 100 people), we do a lookup of this pair in our original celebrity list and see if they should belong to the same cluster (i.e. they have the same category). If they do belong in the same category, and our clustering assigned them to the same cluster, then this pair is a TP. If our clustering assigned them to different clusters, then this pair is a FN. However, suppose we did our pair lookup in our original celebrity file and find that they do not belong in the same cluster (i.e. they have different categories). If our clustering assigned them to the same cluster, then this pair is a FP. If our clustering assigned them to different clusters, then this is a TN.

Precision is then defined as $\frac{TP}{TP+FP}$, and recall is defined as $\frac{TP}{TP+FN}$. Precision can be understood as measuring how correct each of our clusterings are: the percentage of pairs in each cluster that should be grouped together. Recall measures, for each category, how many of the original pairs ended up grouped together in the same cluster.

Based on these definitions, we can see that we would have perfect recall if we had a single cluster (i.e. no false negatives). However, this would yield very low precision. So, for more clusters, we would expect higher precision and lower recall. We also compute the F-value, $\frac{2PR}{P+R}$, which provides us with a more cumulative evaluation of our clustering.

4 Results

Table 2 shows the results for our above clustering techniques. We experimented with a number of different k values (k=4,6,8). For the k-means, we also tried using different starting seeds. We started with random seeds, where we randomly selected k different celebrities to be our starting centroids. Next, we fixed $k = 4$ and tried randomly selecting one celebrity from each of the four groups to be our seeds, which we thought might provide us with better results. We ran 30 trials for each of these k-means experiments and averaged their purity, precision, and recall scores. We also selected four seeds deliberately, one from each of the different categories. This clustering gives us the highest purity, precision and recall scores we have found, although it is possible that there is another set of seeds that maximizes these values. There are approximately ${}_{100}C_k$ possible combinations of seeds, and given the runtime of our k-means clustering algorithm, we were not able to test all of these to find the optimal combination.

We can see that generally, the hcluster algorithm slightly outperforms the k-means algorithm. However, we can see that the performance of the k-means

algorithm highly depends on the seed selection, and if our seeds are chosen well, they can outperform hcluster (e.g. using the personally selected seeds). Additionally, we can see that a higher number of clusters, k, is associated with a higher precision and lower recall, as expected. This relationship is especially evident in the hcluster results. Purity also tends to increase as k increases.

We also have a picture of the hierarchical clustering provided by hcluster. The full picture dendrogram can be found at hcluster.png, and an excerpt is found in figure 1. The hierarchical clustering does not cluster the celebrities into k groups, so we cannot find the precision or recall of the arrangement. However, this dendrogram is the basis for the flat clustering hcluster does, so it is useful to consider in our results.

5 Analysis

Our evaluation techniques described above are a useful way to compare different clustering techniques, but they don't tell us much about what our clusters actually look like. If we carefully examine some of the different clusters that were created using our algorithms, we can see that our system is making some successful (and some unsuccessful) decisions. In order to make our analyses more concrete, we have included two example clusters that were created by our system.

In this section, we will first discuss the results of our k-means clustering, then the hcluster flat clustering, and finally end with a discussion of general issues.

5.1 K-means

Looking at Table 2, we can see that k-means depends heavily on how the initial seeds are generated. Using purely random seeds performs fairly poorly. Its average precision is only .487, meaning that, for any cluster, less than half of the pairs within it should be grouped together. As we increase the number of groups this precision improves, but at the cost of recall, giving us a fairly constant f value. These results suggest that $k = 4$ is just as good as $k = 6$ or $k = 8$ for k-means with completely random seeds. Although the precision goes up as k increases, this is not significant, as the precision will always go up as the clusters have smaller sizes. Precision is more useful for comparing different clusters with the same k, which we will do below.

Although the random k-means result is unsatisfying, the algorithm can be improved greatly using slightly supervised methods to get better seeds. If we pick one seed from each of our four categories, all of our scores increase significantly, especially purity and precision. Again, it is important to note that this is the average of 30 trials, and that some of these seeds do much worse and some do much better. We included the scores for the best group of seeds to show that, given the right starting point, k-means can outperform even hcluster's flat clustering method.

Given this information, the natural question is if we can pick these seeds to maximize the clusters obtained by k-means? Unfortunately there doesn't seem to be a good unsupervised method to do this, as the usefulness of a set of seeds is determined by how alike the seeds are to other celebrities in their respective categories, and conversely how unalike they are to celebrities in other categories. The way we picked the best seeds we found (Kevin Durant, Mariah Carey, Newt Gingrich, and Leonardo DiCaprio) was simply by thinking of celebrities in each category that are not connected to other categories. In general people do not talk about Newt Gingrich in contexts other than political ones, which made him a good seed to get politicians; the same is true about the other seeds. When looking at the clustering produced from these seeds (Table 4), we see that there are very few celebrities that are clustered incorrectly.

One can think of many good methods to pick good seeds with a supervised approach. However this question, while interesting, is outside the scope of our paper. Therefore, we can conclude that, using solely unsupervised approaches, k-means does not give us a satisfying clustering of our data.

5.2 hcluster

The flat clustering method provided by hcluster outperformed the unsupervised k-means algorithm. The way this algorithm performs its clustering is based on the "distance" between different branches of the hierarchical clustering (cophentic distance). To see how the clustering is done, it is useful to look at the dendrogram that hcluster's hierarchical clustering creates (see hcluster.png).

The most important difference between hierarchical clustering and the k-means algorithm is that the hierarchical clustering gives us a more refined view of smaller, sub-clusters of people. For example, consider a snapshot of our hierarchical clustering in Figure 1, which demonstrates how the "sports-like" cluster was created in Table 3. Despite the fact that we have a few unrelated people in this group (e.g. Eminem, 50 Cent, Dwayne Johnson), the groupings are actually fairly sophisticated. As we go down the list of athletes, we can see that athletes of the same sport are even being grouped together (e.g. Kobe Bryant and Lebron James, Andy Murray and Serena Williams). This is a strong advantage of hierarchical clustering, as it provides us a more comprehensive, detailed view of the celebrity relationships that is lost in other flat clustering methods. However, the results of the hierarchical clustering are very qualitative. Thus, it is necessary to create a flat clustering off of this model in order to quantitatively compare it with our k-means results.

The hcluster alrogithm assigns flat clusters to this given hierarchical clustering by identifying significant break points and using this information to best create k specified flat clusters. Generally, the purity and precision/recall scores of these flat clusters are higher than those of the k-means clusters. However, we can see that the assignment of these flat clusters is not perfect (i.e. Cluster 2 in Table 3). The discussion of those errors are discussed in more detail in the General Issues section.

5.3 General Issues

Name similarities ended up having more effect on clustering that we had originally expected. While names are often informative in providing similarities between two celebrities (e.g. if Barack Obama's vector had a high tf*idf score for "michelle"), we also get false positives for clusterings between people like "Rick Perry" and "Tyler Perry" or "Dwayne Wade" and "Dwayne Johnson". These false name-based associations seem to cause issues for both our k-means and hcluster results.

We found that some categories were easier to sort out than others. For example, consider the clustering made in Table 3, which was clustered using the hcluster algorithm with $k = 4$. Clearly, this algorithm had an easier time sorting particular categories over others. For example, Cluster 1 is successfully composed of only politicians. However, Cluster 4 seems to indiscriminately group actors/actresses and musicians together. These results suggest that the tweets used to describe politicans are sufficiently different from the tweets used to describe celebrities in other categories. Contrastingly, these results also suggest that the language used in the tweets to describe musicians and actors/actresses are harder to distinguish than we originally had expected. However, if we consider that most actors/actresses and musicians can be more generally grouped together as today's pop culture icons, it is understandable that twitter users would use similar language to tweet about these celebrities. than specifically refer to their body of work. Distinguishing actors/actresses from artists gets even more complicated when we consider that there are celebrities in our list that cross this barrier (e.g. Justin Timberlake).

However, there were times when some very questionable clusters were created. For example consider Cluster 2 in Table 3. It only contains three people: Nick Swisher, Derek Jeter, and Zach Braff. The grouping of Nick Swisher and Derek Jeter makes sense, since they are both baseball players and they both play for the Yankees; the inclusion of actor Zach Braff seems completely random. However, recall that the number of tweets collected for Zach Braff (4,139) was much lower than the number of tweets collected for other celebrities. In all actuality, there was probably not enough raw data to create a representative or distinguishable vector for Zach Braff, so our algorithm would end up slotting Zach Braff into a nonsensical cluster. This suggests that we might need some sort of baseline for the amount of tweet data we need for a particular celebrity in order to guarantee an adequate clustering.

Lastly, we noticed that our clusters are time sensitive to when we first collected our tweet data and built vectors for each of the celebrities. In other words, the vectors that we build for each celebrity are highly subjective to the time that that our tweet data was collected. For example, when we were first testing our experiments with a small subset of our total data, we noticed that Mitt Romney kept being grouped with Dwight Howard. While this was confusing at first, we realized that the dates that we had collected our data coincided with a popular

GQ Magazine article that named these two celebrities among the "Least Influential People of 2012." This highly-publicized event significantly affected the content of the twitter data that we had collected, leading our system to falsely associate Mitt Romney with Dwight Howard. Thus, if our goal is to find more general associations between celebrities, specific large scale, one time events such as these might adversely affect our dataset.

6 Future Work

For our project, we did a mainly unsupervised approach to our celebrity clustering. Our clustering algorithms had no prior knowledge of which celebrities belonged together, and grouped them solely based off of their tf/idf vector. We did experiment with a supervised technique when we fed our k-means algorithm one seed from each category, and it is arguable that giving the algorithms the number of groups is some kind of supervision as well.

An interesting extension to our experiment, however, would be to do more of a supervised approach. We could have a training set of celebrities that we group ahead of time. Then we could introduce different celebrities into the experiement to see if they get matched with the right cluster. For example, we could have created our four groups of 25 celebrities each first, and then seen if Tom Hanks's tf/idf vector is correctly identified with the "Movies" group. We could evaluate our results based on how many of these celebrities we grouped correctly.

One way to improve our results would be to better clean the raw twitter data. Tweets are notorious for strange abbreviations, misspellings, and altogether for looking very different from the kind of English one would find in a novel or newspaper. We cleaned the corpus by seperating words by whitespace, stripping the words of some punctuation, and putting all the words in lower case. This gave us good results, but it is possible that a better cleaning of the data would improve our results. Go et al propose a number of interesting cleaning methods that could be adapted to our project. In particular, to deal with unigrams that are English words with repeated letters, like "greaaat" and "greaaaaaat", they clean them so that any letter that appears more than two times in a row is replaced with two occurences of that letter. Therefore "greaaat" and "greaaaaaat" would be replaced by "greaat," which would help us group together variants on the same word.

Similarly, our system is limited in that it only looks at unigrams. It would be interesting to look at larger n-grams to see if similar phrases are applied to tweets ("<athlete> played well", "vote for <politician>") and if they improve our results in any way.

One last logical extension would be to change our keywords from names of people in certain categories to other kinds of keywords. For example, instead of having names of athletes, we could have names of sports teams; instead of musicians, have album or song names. It would be interesting to see if our system can group together categories of keywords when names are not involved. Eventually, this could be useful for determining what kind of keyword a search term is. For example, if a user inputs "Lakers," our system could see that the term refers to a sports team and return that information to the user.

7 Conclusions

We hypothesized that by looking solely at Tweets that mention certain celebrities, we can determine which celebrities are similar to each other and what "kind" of public figure they are. In general our experiments supported this assumption, and we found that people use similar words to talk about celebrities who are similar in reality.

We used two clustering algorithms to confirm this, with varying degrees of success. We found that a completely unsupervised approach to k-means clustering, where the seeds are completely random, returns unsatisfying results. However, using marginal amounts of supervision, like ensuring that our seeds come from different categories, improves the quality of the results greatly. Furthermore, it is possible to find seeds which return extremely good clusterings of the celebrities.

We also clustered celebrities using hcluster's flat clustering and hierarchical clustering methods. We found that the flat clustering method was the best unsupervised approach to grouping our celebrities correctly. The hierarchical clustering was also useful to see the kinds of relationships between different celebrities that lead to the final groups.

Overall, we were pleased with the results of the project. Our initial hypothesis, that people tweet about similar kinds of celebrities with similar words and hash tags, was proved correct, and there are many possible extensions to our work that could improve and build upon what we have accomplished in this paper.

8 Acknowledgements

We would like to thank Rich Wicentowski for his help in guiding our project and providing us with information on clustering. We would also like to thank Ameet Soni for his hcluster code, which can be found on his website. Finally, thank you to Sam Clarke '13 for providing us with his twitter scraping code that was essential for our data collection process.

References

[1] Munmun De Choudhury, Nicholas Diakopoulos, and Mor Naaman. 2012. Unfolding the Event Landscape on Twitter: Classification and Exploration of User Categories. In *Proceedings of 2012 ACM Conference on Computer Supported Cooperative Work*

[2] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision. Technical report, Stanford Digital Library Technologies Project.

[3] Cristina Ioana Muntean, Gabriela Andreea Morar, Darie Moldovan 2012. Exploring the Meaning behind Twitter Hashtags through Clustering *Business Information Systems Workshops*, pages 231-242.

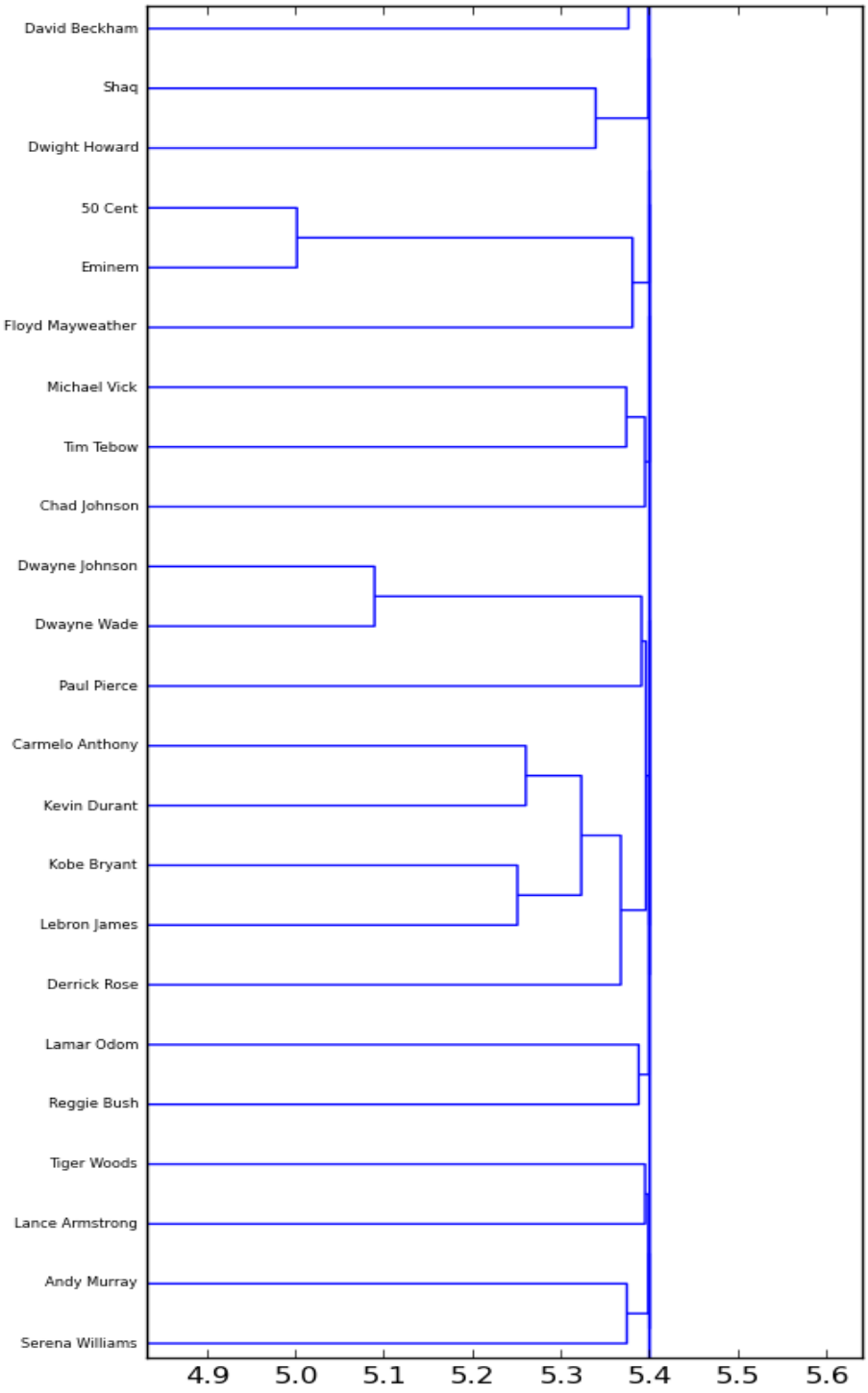

Figure 1: A subset of the hierarchical clustering image created by hcluster

Cluster 1 Barack Obama (P), Al Gore (P), Paul Ryan (P), John McCain (P), Mitt Romney (P), Newt Gingrich (P), Cory Booker (P), Sarah Palin (P), Jerry Brown (P), Mike Huckabee (P), Herman Cain (P), Rick Santorum (P), Ron Paul (P), Gary Johnson (P), Joe Biden (P), Al Franken (P), John Boehner (P), Bill Clinton (P), George Bush (P), Michelle Obama (P), Mitch McConnell (P), Harry Reid (P), Nancy Pelosi (P), Elizabeth Warren (P)
Cluster 2 Nick Swisher (S), Derek Jeter (S), Zach Braff (V)
Cluster 3 Paul Pierce (S), Floyd Mayweather (S), Dwayne Wade (S), Reggie Bush (S), Lamar Odom (S), Wayne Rooney (S), Serena Williams (S), Dwight Howard (S), Lebron James (S), Chad Johnson (S), Lance Armstrong (S), Shaq (S), Kobe Bryant (S), Derrick Rose (S), David Beckham (S), Tiger Woods (S), Andy Murray (S), Kevin Durant (S), Carmelo Anthony (S), Tim Tebow (S), Michael Vick (S), Eminem (M), Snoop Dogg (M), 50 Cent (M), Dwayne Johnson (V)
Cluster 4 Ryan Sheckler (S), Tony Hawk (S), Lady Gaga (M), Justin Bieber (M), Katy Perry (M), Rihanna (M), Britney Spears (M), Taylor Swift (M), Big Sean (M), Nicki Minaj (M), Justin Timberlake (M), Bruno Mars (M), One Direction (M), Adele (M), Alicia Keys (M), Chris Brown (M), Lil Wayne (M), Kanye West (M), Mariah Carey (M), Rick Ross (M), Avril Lavigne (M), Beyonce (M), Jay-Z (M), Ludacris (M), Rick Perry (P), Vin Diesel (V), Jackie Chan (V), Will Smith (V), Adam Sandler (V), Taylor Lautner (V), Tyler Perry (V), Megan Fox (V), Johnny Depp (V), Emma Watson (V), Demi Moore (V), Ashton Kutcher (V), Leighton Meester (V), Morgan Freeman (V), Martin Lawrence (V), Eva Longoria (V), Leonardo DiCaprio (V), Nicole Kidman (V), Will Ferrell (V), Robert DeNiro (V), Natalie Portman (V), Eva Mendes (V), Bruce Willis (V), Christian Bale (V)

Table 3: A sample clustering using hcluster with $k = 4$, with each person's true category listed after their name (P=politics, M=music, S=sports, V=movies)

Cluster 1 Barack Obama (P), Al Gore (P), Paul Ryan (P), John McCain (P), Mitt Romney (P), Newt Gingrich (P), Cory Booker (P), Sarah Palin (P), Jerry Brown (P), Mike Huckabee (P), Herman Cain (P), Rick Santorum (P), Ron Paul (P), Gary Johnson (P), Joe Biden (P), Al Franken (P), Rick Perry (P) John Boehner (P), Bill Clinton (P), George Bush (P), Michelle Obama (P), Mitch McConnell (P), Harry Reid (P), Nancy Pelosi (P), Elizabeth Warren (P)
Cluster 2 Floyd Mayweather(S), Lady Gaga (M), Justin Bieber (M), Eminem (M) Katy Perry (M), Rihanna (M), Britney Spears (M), Taylor Swift (M), One Direction (M), Adele (M), Snoop Dogg (M) Alicia Keys (M), Chris Brown (M), Big Sean (M), Nicki Minaj (M), Justin Timberlake (M), Bruno Mars (M) Lil Wayne (M), Kanye West (M), Mariah Carey (M), 50 Cent (M) Avril Lavigne (M), Beyonce (M), Jay-Z (M), Ludacris (M), Jackie Chan (V) Adam Sandler (M), Tyler Perry (M), Will Ferrell (M)
Cluster 3 Nick Swisher (S), Paul Pierce (S), Dwayne Wade (S), Reggie Bush (S), Lamar Odom (S), Wayne Rooney (S), Dwight Howard (S), Tony Hawk (S), Lebron James (S), Chad Johnson (S), Lance Armstrong (S), Shaq (S), Andy Murray (S), Kevin Durant (S), Carmelo Anthony (S), Tim Tebow (S), Kobe Bryant (S), Derrick Rose (S), Tiger Woods (S), Michael Vick(S), Dwayne Johnson (V)
Cluster 4 Ryan Sheckler (S), Serena Williams (S), David Beckham (S), Derek Jeter (S) Rick Ross (M), Vin Diesel (V), Will Smith (V), Taylor Lautner (V) Megan Fox (V), Johnny Depp (V), Emma Watson (V), Demi Moore (V), Ashton Kutcher (V) Leighton Meester (V), Zach Braff (V), Morgan Freeman (V), Martin Lawrence (V) Eva Longoria (V), Leonardo DiCaprio (V), Nicole Kidman (V), Robert DeNiro (V) Natalie Portman (V), Eva Mendes (V), Bruce Willis (V), Christian Bale (V)

Table 4: A sample clustering using k-means with specially selected seeds: Kevin Durant (S), Mariah Carey (M), Newt Gingrich (P), Leonardo Dicaprio (V). Each person's true category listed after their name (P=politics, M=music, S=sports, V=movies)