Political Party Classification through Document Analysis

Joshua Fan **Irina Tolkova**

Abstract

The overall goal of our project is to classify and position political figures (specifically presidential candidates) based on their statements and speeches. After compiling a collection of speeches, we used tf-idf to extract numerical features from speech texts. To assess the viability of clustering, we used PCA to visualize the data in a lower-dimensional space, and we found that there was some separation between candidates. We then implemented several unsupervised clustering algorithms, including k-means, bisecting k-means, and spectral clustering, and compared their accuracy in classifying documents by candidate (by computing the entropy of each clustering). We found that on a balanced data set, the clustering algorithms (especially spectral clustering) were able to discover clusters of speeches from the same candidate with remarkably high accuracy. On a more unbalanced dataset, the quality of the clusters was worse, but the algorithms were still able to identify many reasonable clusters based on candidate or speech topic.

1 Introduction

Political polarization has become an increasingly pressing issue in American politics. We were interested in exploring how much this has influenced the speeches of presidential candidates in the 2016 election. In particular, given an unlabeled set of speech texts, we wanted to see if an unsupervised clustering algorithm could identify groupings of speeches by candidate, or if speeches with common ideologies and topics could be grouped together.

2 Background

2.1 Tf-IDF

A common method for quantifying textual data for document classification is the term-frequency-inverse-document-frequency method, usually abbreviated as tf-idf. Intuitively, we would like words that help distinguish the document from other texts to have high weight. This implies giving high weight to words that appear often in the document *but do not appear in many other documents*. In 1972, a paper by Jones was the first to develop such an approach which weighed terms within a document so as to represent both relevance within the document, and specificity across the corpus (Jones, 1972).

More formally, let $D = \{d_1, d_2, \ldots, d_N\}$ be a collection of documents, $T = \{t_1, \ldots, t_n\}$ be the set of terms occurring in those documents. Let $f(t_i, d_j)$ be the number of occurrences of term t_i within document d_j. Then, we define the function $tf(t_i, d_j)$ for $i \in [1, n]$ and $j \in [1, N]$ that represents the "term frequency", or measure of relevance of term for a given documents, and the function $idf(t_i, D)$ for $i \in [1, n]$ and that represents the "inverse document frequency", or measure of specificity of term within the corpus. The final weight for a term within a document is

$$tfidf(t_i, d_j, D) = tf(t_i, d_j) * idf(t_i, D)$$

Therefore, the output of tf-idf for a corpus D is a feature matrix X of size $N \times n$, where $X_{ij} = tfidf(t_i, d_j, D)$.

There are multiple weighing schemes that can be used. A simple approach, proposed in the original papers, is to let $tf(t_i, d_j)$ be equal to the number of occurrences (or frequency) of term t_i within document d_j, and to let $idf(t_i, D)$ be equal to the inverse of the number of documents in which term t_i occurs. For this project, we used a different method, with the functions:

$$tf(t_i, d_j) = f(t_i, d_j)$$
$$idf(t_i, D) = log(\frac{N}{|\{d_j \in D | t_i \in d_j\}|})$$

2.2 Dimensionality Reduction

After generating a feature matrix, we were interested in visualizing the data and exploring the potential for clustering. To visualize the data in two dimensions, reduced the dimensionality of the data through Principal Component Analysis (PCA). PCA finds the "best" orthonormal basis for representing the data, such that a reconstruction of the data using projections onto the first k basis vectors has minimal error for all k (Kutz, 2013). Mathematically, given a matrix of observations X, the basis vectors are eigenvectors of the covariance matrix of X, and can be computed efficiently using the singular value decomposition. We used PCA to calculate the coefficients for the first few principal components of the tf-idf feature matrix, and visually evaluated the structure of the data.

2.3 Clustering

Our next focus was clustering. The most prominent (unsupervised) clustering algorithm is k-means (Jain, 2010). This algorithm iteratively assigns each observation to the nearest cluster centroid, and updates each centroid to the mean position of the observations in its cluster. While widely used for many applications, this method has some drawbacks, such as lower performance for high-dimensional data, and assumption of equally-size, spherical clusters. In particular, while k-means attempts to minimize the sum of squared errors from cluster centers, it is only guaranteed to converge to a local minimum (Jain, 2010).

A variant on k-means is bisecting k-means (Steinbach et al, 2000). Iteratively, the data is clustered into two groups using k-means, and one of the two clusters is selected to bisect next. This process repeats until we reach the desired number of clusters. There are multiple measures with which to choose the next cluster to bisect – such as metrics of range or homogeneity. For this project, we simply select the next cluster based on the largest cluster size.

Finally, a third, more sophisticated algorithm is spectral clustering. Spectral clustering transforms the data to a lower-dimensional space before executing a standard clustering algorithm, such as k-means. Intuitively, this transformation is chosen based on the eigenvectors of a matrix representing distances between observations, and the result can also be interpreted as a (relaxed) solution of an optimal cut problem over a graph representing distances between observations. To implement this algorithm, we followed the pseudocode described in (Ng et al, 2001).

After implementing these three clustering algorithms, we would like to compare their performance on our data. We considering evaluating the "internal quality" of the clustering based on the geometry of the resulting clusters, without looking at the labels. However, we wanted to test how well the clusters actually correspond to candidates, so we decided to use candidate labels in evaluating the clusterings.

A commonly-used metric for cluster quality (using class labels) is the entropy of the clustering, which is a measure of the "variation" or homogeneity of each cluster. The entropy of the j-th cluster is defined as:

$$E_j = -\sum_i p_{ij} \log(p_{ij})$$

where p_{ij} is the fraction of data points in cluster j that were from candidate i.

The total entropy is the sum of entropies for each cluster (weighted by cluster size):

$$E_{total} = \sum_{j=1}^{m} \frac{n_j \cdot E_j}{n}$$

where n_j is the number of data points in cluster j, and n is the total number of data points. A small entropy implies that most documents in each cluster came from one particular candidate, which is what we desired. Note that the entropy score is unable to evaluate clusters that were centered around other factors (such as speech topic), but this did not end up being an issue, since all clusters produced by the balanced data set were oriented around one candidate.

3 Data and Implementation

Our data was collected from the American Presidency Project, an online collection of over one-hundred-thousand presidential documents including speeches, interviews, letters, campaign materials, and other texts. For this project, we focused on the 2016 election cycle, and used the web-scraping Python library *lxml* to assemble 332 files from the recent presidential candidates. There is some variation in the style of the document (some are speeches, some are shorter statements or remarks, and some are interviews). Therefore, to standardize our data and reduce word-frequency bias, we decided to use only documents with titles containing the words "speech" or "remarks".

The original dataset had varying numbers of documents from each candidate – for instance, Hillary Clinton is the author of 149 speeches, Donald Trump has 64, and candidates such as Jim Webb have just one – and these documents are of varying length. To simplify the problem, we created balanced subsets of the data to better evaluate clustering methods. For example, to test clustering accuracy, we created a subset with 80 speeches (20 each from Hillary Clinton, Donald Trump, Bernie Sanders, and Rick Santorum the only candidates with at least 20 speeches in the dataset).

We computed tf-idf feature vectors over these datasets using previously described frequency counts for the "tf" measure and the logarithm of the inverse of the fraction of document occurrences for the "idf" measure. Since variation in the lengths of the speeches could bias the tf-idf scores, we also implemented two important optimizations. First, we disregard words that appear in less than 1% or more than 99% of documents, as recommended in (Grimmer & Stewart 2013). This helps ensure that extremely rare or common words do not overly impact the resulting vectors. Secondly, we implemented cosine normalization, which causes all weights to fall into the [0, 1] range, and makes sure that every document is represented by a vector of equal length (Sebastiani, 2002).

PCA was implemented in Python, using the *numpy linalg* library. The three clustering algorithms were also implemented in Python. The centroids were initialized with k randomly-selected points in the dataset. We ran each clustering algorithm 10 times, and picked the clustering with the lowest sum of distances to the nearest cluster center. Then, we compared the entropy scores of each algorithm, and interpreted the physical meaning of the clustering. For spectral clustering, we used the *numpy linalg* library for matrix operations and the *scipy linalg* library to compute the k largest eigenvectors.

4 Results

4.1 TF-IDF

Overall, tf-idf does a good job of returning the most distinctive words of a speech, as intended. Oftentimes, these words reveal important information about the topic or content of the speech. For example, consider the following excerpt (words with highest weight) of a speech's tf-idf vector:

```
[police, policing, bland, sandra, baltimore, justice, violent, mayor,
barbara, deaths, cameras, jordan, reduced, maryland, 1999, improve,
professional, disparities, painfully, sakes, penalty, staffed, cemetery,
city, must, fire, possession, injustice, incarceration, angela, burned,
drug, mental, suppose, criminal]
```

Just from these keywords, we can infer that this speech appears to be about policing, crime, and the criminal justice system. Some specific words also pop up, such as Baltimore and 1999, which give

important contextual information but are less relevant for assessing a speech's thematic content or political ideology.

As another example:

```
[isis, assad, syria, terrorist, arab, sunni, coalition, fly, qaida, zone,
partners, counterterrorism, syrian, saudis, kurds, moderator, fighters,
intelligence, yemen, baghdad, financing, turkey, paris, territory, al,
question, madam, border, refugees, russians, defeat, groups, extremists,
airstrikes, region, resolution, iranians, kurdish, humanitarian]
```

From these high-weight words, it is clear that this speech focuses on foreign policy issues, especially terrorism and conflicts in the Middle East. However, notice that these words alone are not very indicative of a particular political ideology or viewpoint. Thus, as discussed later, k-means tends to cluster speeches about the foreign policy and terrorism together, regardless of the speaker.

4.2 PCA and Data Visualization

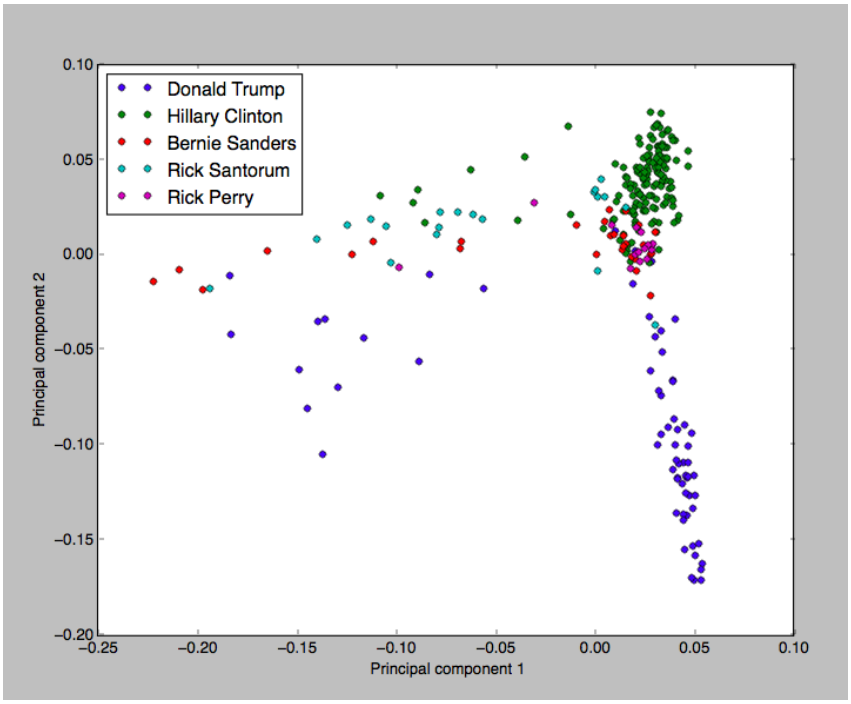

Figure 1: PCA coefficients for components 1 and 2, colored by candidate.

Figure 1 shows the coefficients corresponding to principal components 1 and 2 for the speeches of five candidates (where the foremost component has index 0). Dimension 2 visually captures much of the variation in candidates, and Clinton and Trump appear to be separable across dimension 2. However, the corresponding singular values are low, and show that the two components only account for about 2% of the variance in the data.

4.3 Comparison of clustering methods

The bar graph in Figure (2) shows a comparison of the entropy scores for k-means, bisecting k-means, and spectral clustering. As mentioned in (Steinbach et al, 2000), bisecting k-means produced an improvement in accuracy over regular k-means. This may be because splitting the data into two distinct clusters at each iteration is easier than attempting to identify four clusters simultaneously.

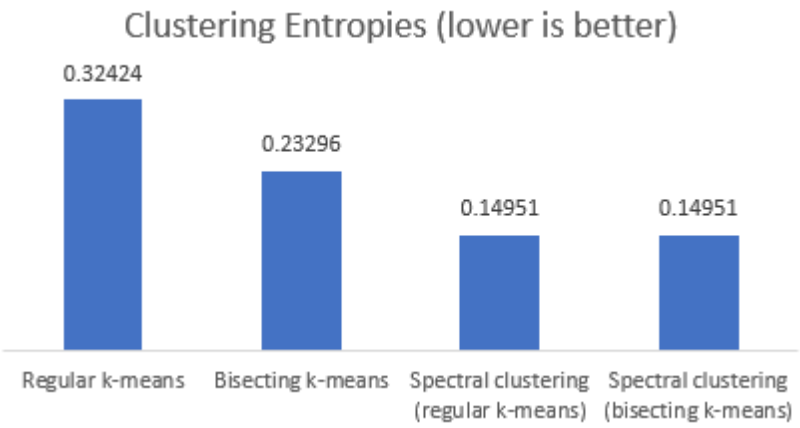

Figure 2: Entropy of clusterings produced by different methods

Spectral clustering resulted in a significant improvement over both regular and bisecting k-means. This speaks to the success of projecting the features to a lower-dimensional space before clustering. In fact, for the two-cluster case, we can plot the data in the two-dimensional spectral space to visualize the separability of the transformed data (see Figure (3)).

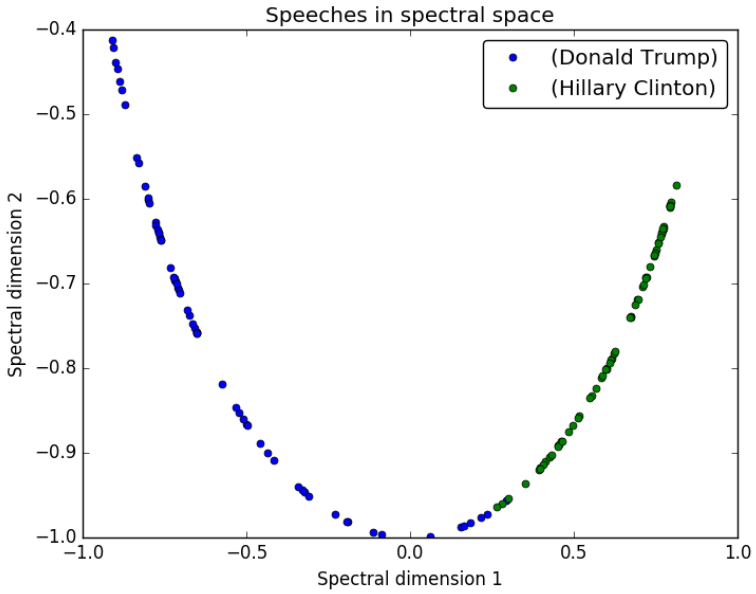

Figure 3: Data transformed into spectral space

4.4 Clustering results

Since we found that spectral clustering resulted in the lowest entropy, we looked more closely at the observations within the four clusters that were formed. First, we could calculate the distribution of candidates falling in each cluster to evaluate the amount of separability by candidate. Additionally, by transforming the cluster centroids back to the space of tf-idf features, we could see their most highly-weighted terms. The results are shown below:

Cluster 0 (95% Donald Trump, 5% Rick Santorum):

[machine, special, nearly, corrupt, immigrant, wikileaks, failed,
navy, refugees, haiti, interests, booing, lied, bill, her,
criminal, factories, enforcement, she, replace, since, renegotiate,
trump, 8th, deported, immigrants, under, plan, deal, mexico, fbi,
jobs, deals, open, administration, borders, nafta, obamacare,
inner, foreign, clintons, immigration, emails, cities, border,
corruption, trade, clinton, illegal, hillary]

Cluster 1 (100% Hillary Clinton):

[commander, still, sure, lot, families, maybe, good, weapons, especially,
friends, matter, clean, black, re, kids, each, believe, reporter, voting,
carolina, lgbt, early, everyone, mother, election, ballot, help, she,
lady, got, tomorrow, everybody, everything, kind, gun, ready, nuclear,
him, stake, khan, think, opponent, tim, vote, know, really, he, florida,
trump, donald]

Cluster 2 (100% Bernie Sanders):

[seniors, decent, delegates, gay, process, pac, billionaires, jail,
millions, almost, morality, pacs, fuel, establishment, israel, major,
justice, tonight, west, wage, corporate, iowa, billionaire, greed,
hampshire, must, fossil, class, banks, income, earth, wall, hour,
health, vermont, bernie, revolution, wealthiest, isis, boos, percent,
political, financial, region, wealth, hayes, super, street, matthews,
sanders]

Cluster 3 (90.4% Rick Santorum, 4.8% Donald Trump, 4.8% Hillary Clinton):

[county, fought, coal, mitt, delivered, values, bella, generation, odds,
society, hampshire, grandfather, document, unidentified, bless, oil,
abortion, gingrich, rights, party, iowa, believed, tonight, government,
laughter, little, god, reince, message, health, declaration, obamacare,
conservative, missouri, barack, rick,race, folks, town, gentlemen,
applpause, santorum, somerset, ladies, romney, pennsylvania, someone,
freedom, karen, cheers]

These clusters all corresponded very well to a particular candidate: only 3 out of the 80 speeches were not placed in the "correct" cluster. Looking at the words with the highest weight for each cluster centroid is quite informative, since they match up with common topics and themes that candidates frequently talked about. For example, the words in Cluster 0 referenced major themes brought up by the Trump campaign (illegal immigration, refugees, trade, border control, etc.) as well as lines of attack used (such as Hillary Clinton's email scandal, special interests and corruption, attacking Obamacare, Wikileaks, etc.).

5 Conclusion

While k-means is a very commonly used algorithm, there are few convergence guarantees. In fact, for calculating k clusters over n data points of dimension d, its worst-case runtime is $O(n^{kd})$, with some tighter bounds developed for special cases (Inaba et al, 1994; Dasgupta, 2003; Arthur and Vasilvitskii, 2006). For this project, this becomes a spectacularly bad guarantee, since $d > 8000$ for tf-idf features. However, in practice, the algorithm tends to run very quickly. As a result, the extra computations in other clustering algorithms – specifically, in spectral clustering – are likely to extend the running time of the algorithm.

Bisecting k-means is built on *k-1* iterations of k-means with a cluster size of 2. Therefore, its worst-case runtime complexity is $O(k * n^d)$, though it is considered to be efficient in practice (Steinbach et al, 2000).

On the other hand, spectral clustering has $O(n^3)$ operations added to the expense of clustering: it requires multiplying several $n \times n$ matrices and computing the k largest eigenvectors, all of which

take $O(n^3)$ time (Yan et al, 2009). In practice, these $O(n^3)$ operations might outweigh the runtime of k-means, which is executed after the matrix computations and data transformation. So, although spectral clustering performed the best out of all these methods, it could run into scalability issues if the size of the dataset was increased significantly.

All in all, it was fascinating that an unsupervised method could cluster campaign speeches by candidate with high accuracy, simply based on word frequencies (tf-idf scores). This indicates that candidates and political ideologies often have distinctive "signatures" of words they use frequently or topics that they mention. Clustering performance does worsen when the number of speeches per candidate is unbalanced; nevertheless, the clustering algorithms were still able to find some clusters corresponding to certain candidates or topics (such as grouping foreign-policy or economics speeches together, regardless of candidate).

6 References

Arthur, D., & Vassilvitskii, S. (2006). How slow is the k-means method? *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, 144-153.

Dasgupta, S. (2003). How fast is k-means? *COLT Computational Learning Theory*. **2777**: 735.

Grimmer, J. & Stewart, B. (2013) Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis* **21**(3):267-297.

Inaba, M., Katoh, N. and Imai, H. (1994). Applications of weighted voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). *SCG '94: Proceedings of the tenth annual symposium on Computational geometry*. 332–339.

Jain, A. (2010) Data clustering: 50 years beyond K-means. *Pattern Recogition Letters* **31**:651-666 .

Jones, K. (1972) A statistical interpretation of term specificity and its application in retrieval. *Journal of Documents* **28**(1):11-21.

Kutz, J. (2013). Data-Driven Modeling & Scientific Computation : Methods for Complex Systems & Big Data (First ed.). Oxford: Oxford University Press.

Ng, A., Jordan, M. & Weiss, Y. (2001) On Spectral Clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems* **14**:849-856.

Sebastiani, F. (2002) Machine Learning in Automated Text Categorization. *ACM Computing Surveys* **34**(1):1–47.

Steinbach, M. et. al. (2000) A Comparison of Document Clustering Techniques. *KDD Workshop on Text Mining.*

Yan, D., Huang, L. & Jordan, M. (2009) Fast Approximate Spectral Clustering. *15th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD)*: 907-916.