Intermediate Report

Nick Porter, Christian Rachmaninoff, Madhur Pandey

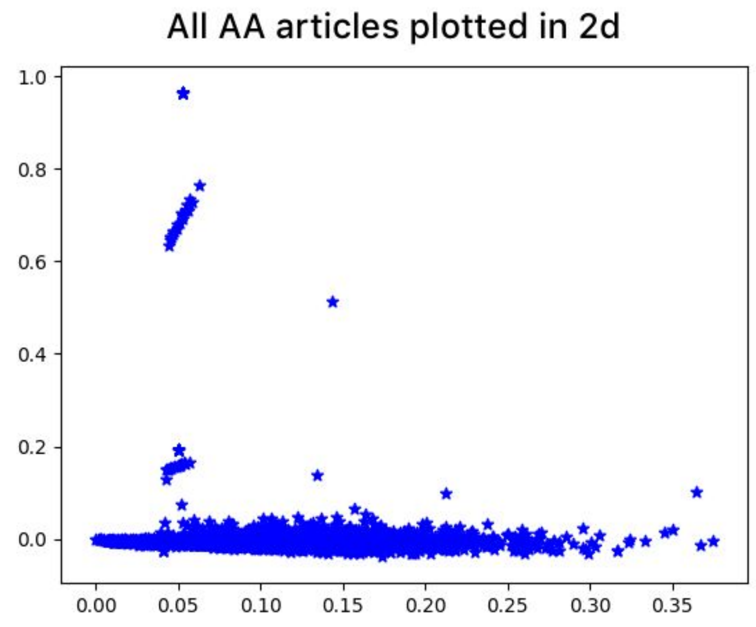
For our data collection report, we had extracted plain text files for each Wikipedia article. Our first goal was to transfer documents these documents into another data representation more conducive to analysis.

Initially, We chose to create a sparse vector for each article with each index corresponding to a boolean value indicating that a specific trigram is present in the document. We hoped that this would allow us to calculate article similarity/distance, which in turn would allow us to potentially cluster articles by category. To convert the articles into vector format, we generated a set of trigrams present in each article after removing stop words. We then created an index of all trigrams in the entire corpus, allowing us to map a specific trigram to an index in one of the document vectors. We then went back through the trigram sets and mapped each trigram to its index and generated the sparse vectors. We then attempted to run PCA on the document to trigram matrix.

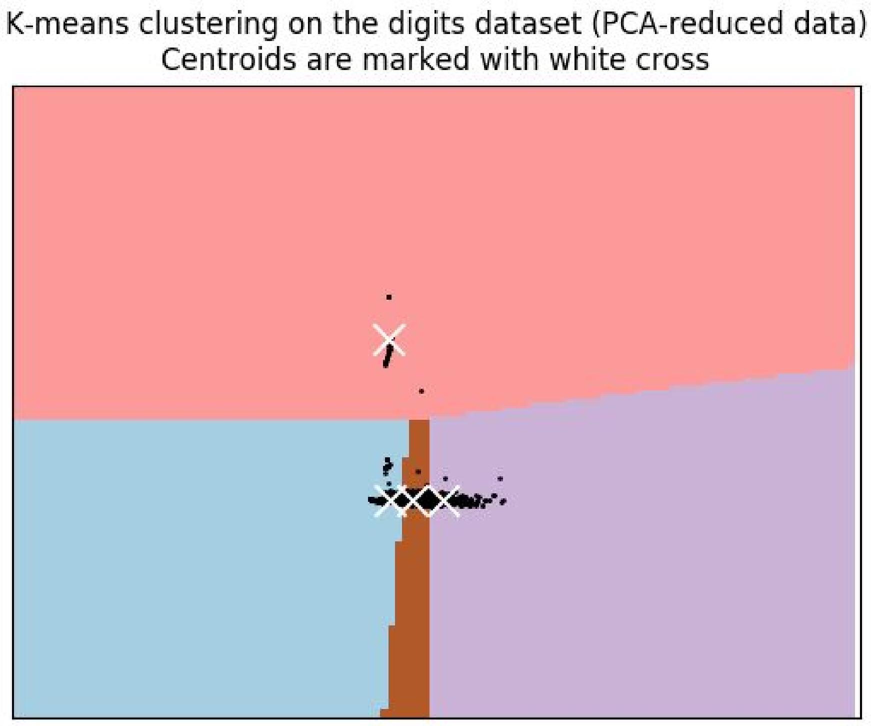
Unfortunately we had limited success with this approach, the vectors were too large and caused our system to run out of memory at runtime. We may re-attempt this approach using Spark or a similar distributed computing framework in hopes that memory constraints would not be an issue in a cluster computing environment. Consequently, we chose to look other possible document representations. We used a vectorizer built into SKLearn to convert our collection of raw text Wikipedia Articles into a matrix of TF-IDF features. When vectorizing we removed a list of english stop words and we used inverse-document-frequency reweighting to give more/less weight to certain terms.

Using the generated vectors we ran a version of PCA called TruncatedSVD which deals well with sparse matrices. We took the first two principal components for each data point, reducing our data down to two dimensions. This allowed us to more easily plot the data points to visualize the structure, and determine if there were any clearly visible clusters.

Below is our graph of all the wikipedia articles that start with ‘AA’. As you can see the documents seem to cluster. We expect that when we expand these algorithms to all of our data we will see more interesting clusters.



We also ran K-means clustering on all the ‘AA’ documents.



K-means++ was used to initialize the centers and we set k = 4.
Above you can see the graph has one very central cluster and because of this the data for the ‘AA’ articles doesn’t cluster very well.

We wanted to understand more than just the structure, so we looked at which terms appeared often in each cluster. When k=4 we found these to be some of the popular terms in the clusters.

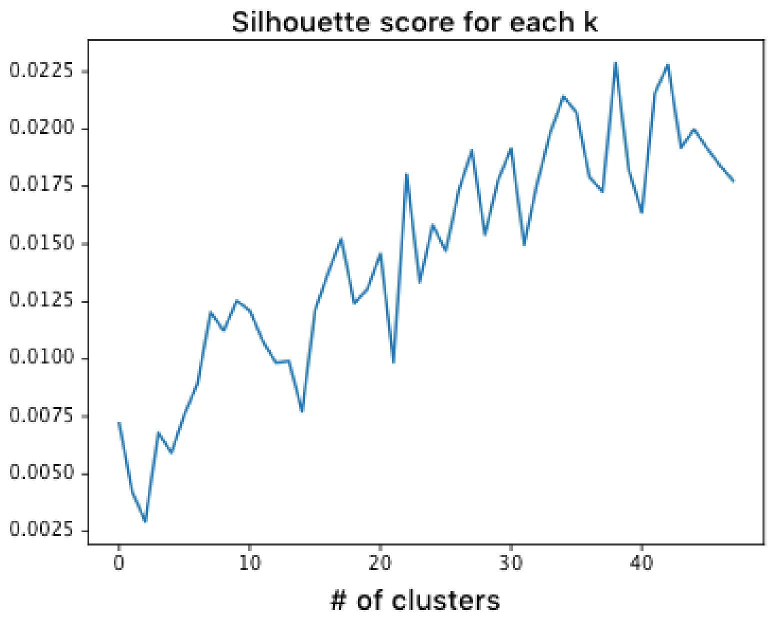
```
Cluster 0:
 refer, disambiguation, alfonso, alexander, albert, ii, asa, iii, cow,
anastasius

Cluster 1:
 game, games, league, player, team, players, baseball, season, atari, ball

Cluster 2:
 april, city, government, century, world, war, church, country, people,
population

Cluster 3:
 august, computer, function, software, data, example, theory, carbon, energy,
cell
```

Determining the correct k value when running k-means plays a large role in the quality of the clusters generated, so we clustered all the ‘AA’ articles from [2-50] clusters and plotted the silhouette score for each. The plot was not that informative and it was a computationally expensive task that we may or may not run on our complete dataset.


Silhouette score for each k
# of clusters