CS 6001 Applied Spatial and Temporal Data Analysis - Spring 2017 - Homework 3

Dalton Cole

March 13, 2017

1 Data Retrieval

For this assignment, I had to retrive 100 CNN articles and convert them to a data matrix. This is done using the **scrapy.py** script, which is a python script. This script grabs the articles and title data from 100 CNN articles that have their URLs in **website_list**. The articles must have at least 150 words to be considered a valid article. The articles used are in **used_site_list.txt**. For this experiment, all URLs in **website_list** were used.

Five different categories of articles were used: Politics, Technology, Investment, Travel, and Health. These articles range over a wide time period but are about roughly the same subject (in respect to their categories).

The python script creates a data matrix with the article data. It counts the frequency of each word in each article and saves it in a csv file. The type of article is also stored in this file. The data is saved to **data.csv**.

2 Classification

Next, I had to run K-means clustering on the data matrix. K-means is explained in Algorithm 1. The script used to run this is **k_means.py**. The script takes the **data.csv** file and sorts the data into articles and article type. I then do K-means clustering using Euclidean, Cosine, and Jarcard distances. The nltk python module is used. Nltk's Euclidean and Cosine distances are used, while Jarcard is implemented in **jarcard_similarity.py**.

Algorithm 1: Algorithm for K-Means

```
1 Select K points as the initial centroids
2 Repeat
3 Form K clusters by assigning all points to the closest centroid
4 Recompute the centroid of each cluster
5 Until The centroids don't change
```

After this, Sum of Squared Error (SSE) is computed through the function in **sum_of_squares.py**. This is done by averaging the word frequency of every word in every cluster. Each data point in each cluster subtracts the average from itself, squares this value, and sums each value up. This equation is shown in Equation 1.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y})^2 \qquad (1)$$

3 Results

Tables 1, 2, and 3 show the SSE and Percent Correctly Clustered Together for each distance over three different trials. Table 4 shows the average percent of success rate over each trial. Tables 5, 6, and 7 show the same information but for when feature reduction is applied to the data set before K-means is applied. Table 8 shows the average percent of success rate over each trail for the feature reduced data.

Distance Formula	SSE	Percent Correctly Clustered Together
Euclidean	116604	37%
Cosine	119612	49%
Jacard	119434	46%

Table 1: Correctness of each Distance without Feature Reduction, Trail 1

Distance Formula	SSE	Percent Correctly Clustered Together
Euclidean	117072	53%
Cosine	121698	50%
Jacard	117167	55%

Table 2: Correctness of each Distance without Feature Reduction, Trail 2

Distance Formula	SSE	Percent Correctly Clustered Together
Euclidean	120442	23%
Cosine	120267	53%
Jacard	117446	90%

Table 3: Correctness of each Distance without Feature Reduction, Trail 3

Distance Formula	Percent Correctly Clustered Together Average
Euclidean	37.67%
Cosine	50.67%
Jacard	63.67%

Table 4: Average Percent Correct, without Feature Reduction

Distance Formula	SSE	Percent Correctly Clustered Together
Euclidean	86966	55%
Cosine	91071	84%
Jacard	90106	87%

Table 5: Correctness of each Distance with Feature Reduction, Trail 1

Distance Formula	SSE	Percent Correctly Clustered Together
Euclidean	86814	49%
Cosine	89783	90%
Jacard	93848	49%

Table 6: Correctness of each Distance with Feature Reduction, Trail 2

Distance Formula	SSE	Percent Correctly Clustered Together
Euclidean	87128	54%
Cosine	90271	89%
Jacard	89998	58%

Table 7: Correctness of each Distance with Feature Reduction, Trail 3

Distance Formula	Percent Correctly Clustered Together Average
Euclidean	52.67%
Cosine	87.67%
Jacard	64.67%

Table 8: Average Percent Correct, with Feature Reduction

4 Conclusion

From previous assignments, it was expected that Cosine would out perform both Euclidean and Jacard distances. From the non-feature reduced data set, this is not seen. In fact, Jacard performed the best, followed by Cosine, and then Euclidean in last. All three of these distance functions performed poorly. With feature reduction however, Cosine did out perform the others by a decent margin. This is likely because Cosine deals with higher dimension data better than the other two distances. A possible explanation for feature reduction being better than the non-feature reduction set is that there were to many zeros in the data set. This made the data have very low variation, causing each distance metric to perform poorly. By having only attributes with a 80% or greater variance, then articles with litter variation from other clustered articles appear less similar. Only the major differences between articles remain.