Applied Spatial and Temporal Data Analysis

Vijay K. Shah

I. Data Retrieval

Q 1. Find 100 CNN news articles online (try to find them in different categories, e.g., sports and finance). You need to find the new article by yourself. Pls ignore picture or other non-text data in the new article.

In this section, I first describe how 100 CNN News links were retrieved followed by the discussion on data retrieval from each link. **Note:** I actually crawled 200 CNN News articles for this assignment.

A. *100 CNN News Link Retrieval:*

Step 1: I downloaded the urls for all the articles in the webpage - http://www.cnn.com/US/archive/
Step 2: I stored them in a text file **links.txt** in home folder **helloProject**.
Step 1 and 2 are handled by script file **cnn_spider.py** stored in folder **helloProject/helloProject/Spiders/**

B. *Data (Articles) Retrieval from all links:*

I created a script **pages_spider.py** to navigate through all urls (stored in links.txt) and download the content body using Scrappy request. Here, I used Scrapy selector.xpath('//p/text()') , selector.xpath('//h/text()') and selector.xpath('//span/text()') so that only useful texts are downloaded instead of entire html file. After that, I only consider alphabetical words with atleast 3 alphabets using a regrex [a-zA-Z]$\{3,\}$

Note: The script **pages_spider.py** can be found in folder "helloProject/helloProject/Spiders" Each downloaded article is stored as article- **article_%Article-headerName%.txt** in folder **Articles/**

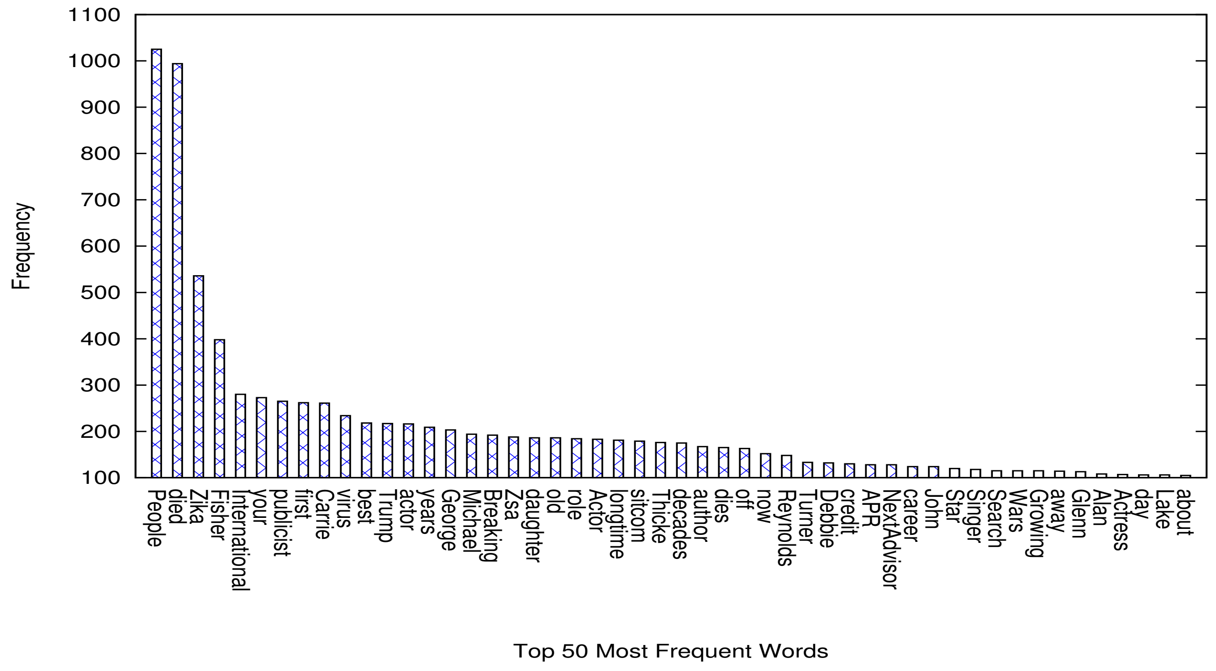

Fig. 1: Frequency vs Top K features (most frequent)

II. Feature Selection

Q 2. Convert them to data matrix (each row is an article and each column is a unique term).

In this section, I present the details of forming a data matrix where each row is an article and each column is a unique word.

A. *Creation of data matrix*

Step 1: I filtered out insignificant words from the data retrieved from each article as discussed in Homework 1.
Step 2: I sorted out the list of words in decreasing order of its frequency. (only until I get first top K frequent unique words).
Step 3: I created a data matrix of size (Number of Articles) X (Top K Most Frequent Words). Each cell contains the frequency of each unique word (Column) for following article (Row).

B. *Addition of Category Column to the data matrix*

I have added a new column called *Category*, which represents the category of the article (row). The category list is as follows:

- **entertainment**
- **us**
- **health**
- **politics**
- **opinions**

III. Clustering with different Similarity Metrics

Q 3. Run K-means clustering with Euclidean, Cosine and Jaccard similarity. (Specify K as the number of categories of your 100 CNN news articles)

A. *K-Means with Euclidean Distance*

We have run K-means clustering with Euclidean Distance taking (i) 3 features and (ii) 9 features as shown in Fig. 2 and Table I respectively.

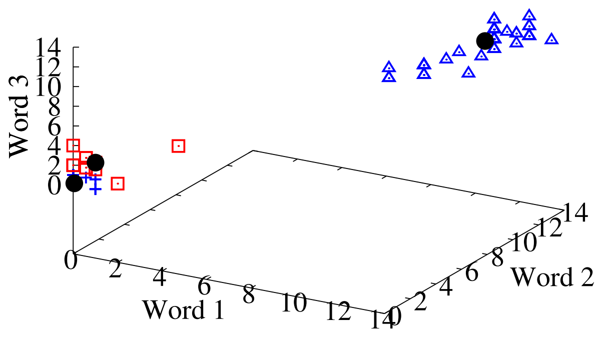

Fig. 2: Euclidean Distance (with 3 features)

Centroids	lost	People	died	Zika	Fisher	International	your	publicist	first	No. Articles
centroid0	0.09565	0.04348	0.30435	4.66087	0.14783	1.88696	2.25217	0	0.82609	115
centroid1	11.79104	11.73134	11.13433	0	3.16418	0	0.13433	3.0597	1.8806	67
centroid3	11.7	11.55	10.35	0	3.65	3	0.2	3	1.9	20
centroid4	0	1	1.33333	0	32	1	0.33333	0	1	3

Table I: Euclidean Distance (with 9 features)

B. *K-Means with Euclidean Similarity*

We have run K-means clustering with Euclidean Similarity taking (i) 3 features and (ii) 9 features as shown in Fig. 3 and Table II respectively.

Computation of Euclidean Similarity: For a pair of articles say, Article(i) = $\{w_{i0}, w_{i1}, \ldots w_{i(K-1)}\}$ and Article(j) = $\{w_{j0}, w_{j1}, \ldots w_{j(K-1)}\}$, the euclidean similarity $E(ij)$ is given by

$$E(ij) = \frac{1}{1 + \sqrt{(Article(i) - Article(j))^2}} \tag{1}$$

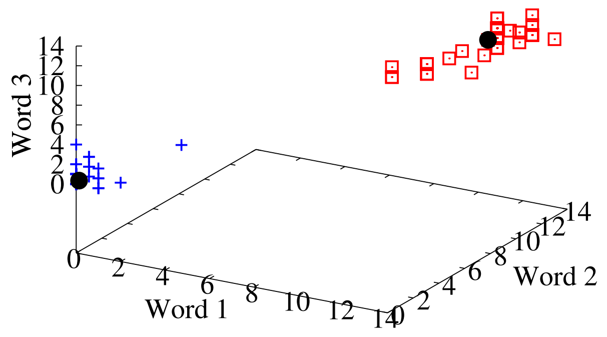

Fig. 3: Euclidean Similarity (with 3 features)

Centroids	lost	People	died	Zika	Fisher	International	your	publicist	first	No. Articles
centroid0	0	0.13636	0.22727	22.72727	4.36364	3.04545	2.09091	0	1.68182	22
centroid1	5.65574	5.5847	5.39344	0.19672	1.65027	1.16393	1.24044	1.44809	1.22951	183

Table II: Euclidean Similarity (with 9 features)

C. K-Means with Cosine Similarity

We have run K-means clustering with Cosine Similarity taking (i) 3 features and (ii) 9 features as shown in Fig. 4 and Table III respectively.

Computation of Cosine Similarity: For a pair of articles say, Article(i) = $\{w_{i0}, w_{i1}, \dots w_{i(K-1)}\}$ and Article(j) = $\{w_{j0}, w_{j1}, \dots w_{j(K-1)}\}$ the cosine similarity $C(ij)$ is given by

$$E(ij) = \frac{Article(i)Article(j)}{||Article(i)||.||Article(j)||} \tag{2}$$

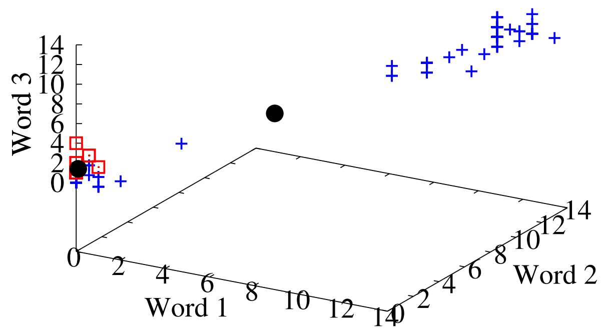

Fig. 4: Cosine Similarity (with 3 features)

Centroids	lost	People	died	Zika	Fisher	International	your	publicist	first	No. Articles
centroid0	9.89423	9.84615	9.31731	0	3.78846	0.60577	0.15385	2.54808	1.67308	104
centroid2	0.04545	0	0.54545	3.54545	0	2.36364	0	0	1.86364	22
centroid3	0.01639	0.01639	0.06557	7.45902	0	2.70492	3.13115	0	0.77049	61
centroid4	0.22222	0	0.38889	0.16667	0.22222	0	3.66667	0	0	18

Table III: Cosine Similarity (with 9 features)

D. K-Means with Jaccard Similarity

We have run K-means clustering with Jaccard Similarity taking (i) 3 features and (ii) 9 features as shown in Fig. 5 and Table IV respectively.

Computation of Jaccard Similarity: For a pair of articles say, Article(i) = $\{w_{i0}, w_{i1}, \ldots w_{i(K-1)}\}$ and Article(j) = $\{w_{j0}, w_{j1}, \ldots w_{j(K-1)}\}$, the jaccard similarity $J(ij)$ is given by

$$J(ij) = \frac{Article(i) \cap Article(j)}{Article(i) \cup Article(j)} \tag{3}$$

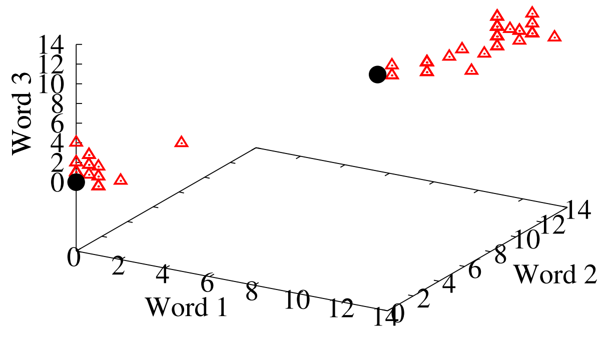

Fig. 5: Jaccard Similarity (with 3 features)

Centroids	lost	People	died	Zika	Fisher	International	your	publicist	first	No. Articles
centroid0	0.0566	0.00943	0.22642	5.0566	0.03774	2.04717	2.4434	0	0.86792	106
centroid3	10.39394	10.34343	9.77778	0	3.9798	0.63636	0.14141	2.67677	1.71717	99

Table IV: Jaccard Similarity (with 9 features)

IV. Evaluation of K-means with SSE

Q 4. Evaluate K-means clustering results with SSE

We evaluate K-means clustering against different metrics with SSE for following cases (i) 3 features (ii) 9 features and (iii) 50 features.

Metrics	SSE (3 features)	SSE (9 features)	SSE (50 features)
Euclidean Distance	270.074074074	13279.4824897	14386.9788434(5 clusters)
Euclidean Similarity	133.046	2.077	1.425 (2 clusters)
Cosine Similarity	inf	inf	149.869 (4 clusters)
Jaccard Similarity	inf	103.001	101.11 (4 clusters)

Table V: Evaluation with K-Means against different Metrics

V. Analysis

As aforementioned, there are basically 2 clusters for dataset with 3 features, detected by all of above similarity metrics. Whereas with 9 features, Euclidean distance is able to detect upto 4 clusters (close to original 5 clusters (categories)). However, with 50 features, all 5 clusters were determined by euclidean distance. The cosine and jaccard similarity metrics also yield upto 4 clusters. Among similarity metrics, the SSE is least for euclidean similarity.

However, it is evident that the euclidean distance performs the clustering best. It may be because the distance between any two point is significant in case of euclidean distance, hence easily distinguishable to be assigned to a certain closest centroid. Also, it requires least number of iterations to converge compared to other metrics (i.e. similarity metrics).