Name: Smitesh Patil

Student Id: 22223696

Assignment 1

1. In class, we briefly discussed pre-processing techniques such as stemming, stop-word removal, and thesaurus construction. Given a text document, suggest any three additional pre-processing techniques that may be used. Explain the approach and outline the potential benefit of the approach.

Answer: Numerous other techniques could be used to pre-process data such that it is easy to retrieve

- Dimensionality Reduction: In this pre-processing technique, we are concerned with reducing the number of features in our system. When a system has a lot of data with a lot of features it affects the system performance later, this phenomenon is called the curse of dimensionality. To tackle this problem, we need to reduce the number of features or "dimensions" and maintain variation in data.

  The most widely used dimensionality reduction technique is called Principal Component Analysis. In this technique, we try to transform the data into a new coordinate system based on the initial data such that the variation of the initial data stays the same.

- Feature Engineering/Selection:

  Suppose in your information system you have housing price data stored with various dimensions like age of the house, locality, length of the house plot, width, value, etc. We can make out over here that the length and width of the house plot are related to the size hence we can combine the length and width dimensions into just one dimension area = length * width. In this technique, we try to create better features that would help us create a better system.

- Handling missing values: In many retrieval systems, you can find data that has missing values like NA, NaN, or NULL. You can handle such values using various statistical measures like mean, and median or assigning them minimum or maximum values from the dataset.

2. Given the following small sample document collection:

(a) D1: Shipment of gold damaged in a fire

(b) D2: Delivery of silver arrived in a silver truck

(c) D3: Shipment of gold arrived in a truck

Calculate the term weightings for terms in D1. Show your workings and state any assumptions you make. (10 marks)

Answer:

Steps taken to generate term weighting for terms in D1.

- Remove stop words from all the documents as they don't add much. And use stemming to get to the root word.
  (a) D1: Shipment gold damage fire
  (b) D2: Delivery silver arrive silver truck
  (c) D3: Shipment gold arrive truck

- Get the frequency of terms in the document: D1: Shipment gold damage fire

  Total number of terms in document D1: 4

  Term Frequencies:

  $$Shipment = \frac{Number\ of\ time\ shipment\ occurs\ in\ document\ D1}{Total\ number\ of\ terms\ in\ document\ D1} = \frac{1}{4}$$

  $Gold = \frac{1}{4}$ $Fire = \frac{1}{4}$

  $Damage = \frac{1}{4}$ $Delivery = \frac{0}{4}$

  $Silver = \frac{0}{4}$ $Arrive = \frac{0}{4}$

  $Truck = \frac{0}{4}$

  Inverse Document Frequencies:

  $$Shipment = Log\{\frac{Total\ number\ of\ documents}{Total\ number\ documents\ with\ term\ shipment\ in\ it}\} = \log\{\frac{3}{2}\} = 0.585$$

  $Gold = log\frac{3}{2} = 0.585$ $Fire = log\frac{3}{1} = 1.585$

  $Damage = log\frac{3}{1} = 0.585$ $Delivery = log\frac{3}{1} = 1.585$

  $Silver = log\frac{3}{1} = 1.585$ $Arrive = log\frac{3}{2} = 0.585$

  $Truck = log\frac{3}{2} = 0.585$

- Use the calculated weights and apply them to terms in D1

  D1: Shipment of gold damaged in a fire
  Remove stop words and apply the stemming technique.
  D1: Shipment gold damage fire

  Term 1: $Shipment = \frac{1}{4} * 0.585 = 0.14625$
  Term 2: $Gold = \frac{1}{4} * 0.585 = 0.14625$
  Term 3: $Damage = \frac{1}{4} * 1.585 = 0.39625$
  Term 4: $Fire = \frac{1}{4} * 1.585 = 0.39625$

- In class, we discussed the document collection as term-document matrix, where each cell in the matrix indicates the usefulness of term i in describing document j. We also discussed how we could evaluate the similarity of a query and document.
  Outline a suitable indexing structure to store the information in the matrix (note that matrix is sparse). (10 marks)
  Outline at a high level, in pseudo-code, an algorithm to calculate the similarity of a document to a query. (10 marks)

Answer:

Outline a suitable indexing structure to store the information in the matrix (note that matrix is sparse). (10 marks)

Sparse matrices are defined as matrices that have predominantly zero values with a few non-zero values in between. Storing a sparse matrix in a way similar to a traditional matrix causes a lot of redundancy as zero values are insignificant.

We can store such a matrix in a form of a list that contains the row, the column, and the value of the term t_i. While retrieving the matrix for querying purposes we can populate a regular matrix by using data from the condensed matrix with non-zero values by taking the rest of the absent data as zero.

Outline at a high level, in pseudo-code, an algorithm to calculate the similarity of a document to a query. (10 marks)

$$Term\ Frequencey = \frac{Number\ of\ time\ term\ occurs\ in\ document}{Total\ no\ of\ terms\ in\ document}$$

$$Inverse\ Document\ Frequencey = \log\{\frac{Total\ no\ of\ documents}{No\ of\ documents\ with\ term\ in\ it}\}$$

$$Cosine\ Similarity = \frac{A*B}{|A||B|}$$

Procedure CalculateDocumentSimilarity(Documents, Q)

```
tf <- term frequency of all words in the documents
idf <- inverse document frequency of all the words in the documents
cosine_similarity_arrary <- empty array that stores documents similar to user query Q
For index, document in documents do
        cosine_similarity_arrary[index] <- cosine_similarity(tf*idf of the document, Q)
For stop
return(cosine_similarity_array)
```

In the above pseudo-code, we first calculate the term frequencies and inverse document frequencies of all the documents and store them in matrixes then we loop through the matrix and apply cosine similarity to calculate the similarity between the document and the user query Q. We store it in an array with the index corresponding to the document. Thus, we have an array with similarity calculated for the documents which can be sorted to retrieve documents with the highest similarity to the user query.