Name: Sahil C | Ashutosh W
Class: MSc. Data Analytics
Student ID:
23100637 | 23100703

Information Retrieval
Assignment 1

1 Question 1

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

Ans. When the matrix is sparse, the **Inverted Index approach** can be used to address the problem. Let us consider a corpus of 10,000 words. If we use the Term Document Index, we must create a matrix of 10,000 x 10,000 cells, and if we assume that each cell takes 0.1 byte of space, the total space to store the variables will be 0.1x10^12, which is approximately 100Gb, and this is only to store the variables. When we look at this matrix, we can see that the majority of the values are 0 and take up around 85% of the space, which is useless. This can be solved using the *Inverted Term matrix*, which generates a dynamic array of components that only contains the locations where the variables are present, so saving space and increasing the retrieval system's efficiency. The following are some of the benefits of Inverted Term Indexing:

1. **Efficient Storage**: This method is space-efficient because it only stores information about terms that actually occur inside the document.
2. **Fast Query Processing**: As it saves the term which occur in the document the retrieval is much more faster than other methods.
3. **Reduced Memory Usage**: Storing the 0 values can take a lot of memory so instead of saving the 0, this method significantly reduces the amount of memory required to save the same information

This method is like a **Hashmap** data structure, which directs you from a word to a document or a web page.

To carry out the Inverted Term Indexing there are few steps which needs to be followed to get the output. We can consider these steps as psuedo alogorithm for the Inverted Term Matrix. Lets understand the algorithm with an example:
Let the ***corpus*** = **Doc 1(a,b,c,d), Doc 2(b,e,f,c)**.

1. **Token Sequencer**: In this step, sequence of each term in each document is given the document ID.

Step 1 : Token Sequencer	
Term	**Doc ID**
a	A
b	A
c	A
d	A
b	B
e	B
f	B
c	B

2. **Sorting**: Once the tokenization is completed then we sort the whole list in ascending order.

Step2 : Sorting	
Term	**Doc ID**
a	A
b	A
b	B
c	A
c	B
d	B
e	B
f	B

3. **Grouping**: The next step is to group the term by words and then give the document ID.

Step 3 : Grouping	
Term	**Doc Frequency**
a	1
b	2
c	2
d	1
e	1
f	1

4. **Splitting it into dictionary and posting**: Once the Step 3 is completed we get the complete dictionary, in this step we can point to the posting list for each term.

Step 4 : Posting List					
Term	**Doc Frequency**	**Posting List**			
a	1	→	A		
b	2	→	A	→	B
c	2	→	A	→	B
d	1	→	A		
e	1	→	B		
f	1	→	B		

5. **Calculate the frequency**: Document frequency information is added.
6. **Calculate Similarity Score**: Using the similarity measures like Cosine Similarity to compare the *TF-IDF* vectors of the query and the document.
7. **Evaluate the Score**: Resulting similarity score indicates how much that document is to the query.

   Now, in order to retrieve a certain term we can use a *B+ tree* in collaboration with the posting list which is generated. We can build a *B+* tree that behaves as a secondary index to the posting list. The keys which are in the *B+* tree will correlate with the terms and the values will be the pointer to the posting list. Now when a query is issued the *B+* tree can quickly locate the posting list associated with the terms.

2 Question 2

With respect to D1 = Shipment of gold damage in a fire and a query = gold silver truck, consider how the similarity sim(Q, D1), *should* change for each of the following augmentations to D1.

1. **D1 = Shipment of gold damaged in a fire. Fire.**
2. **D1 = Shipment of gold damaged in a fire. Fire. Fire.**
3. **D1 = Shipment of gold damaged in a fire. Gold.**
4. **D1 = Shipment of gold damaged in a fire. Gold. Gold.**

Note, there is no need to show any calculations; the question pertains to how the similarity should change.
Ans.
There are various approaches for determining the similarity between the query and the documents, one of them is the **Boolean Space model** and **Vector Space model**. Both approaches determine how relevant the query is to the given document.

1. **D1 = *Shipment of gold damaged in a fire. Fire.***:- In this case, adding "Fire" to the D1 will reduce the relevance of the document being retrieved, as the word "Fire" will be added to the vector which will increase the N(Total number of words) and computationally it will decrease the score.
2. **D1 = *Shipment of gold damaged in a fire. Fire. Fire.***:- Similarly, in this scenario, adding "Fire", "Fire" to the D1 will further reduce the relevance of the document being retrieved, as the word "Fire" "Fire" will be added to the vector which will again increase the N(Total number of words) and eventually it will decrease the score. Hence, the similarity score for "Fire", "Fire" will be further reduces.
3. **D1 = *Shipment of gold damaged in a fire. Gold.***:- But when term "Gold" is added to the document it is matching the query and will increase the similarity score of the document as compared to the first two result.
4. **D1 = *Shipment of gold damaged in a fire. Gold. Gold.***:- Similarly when the "Gold.Gold" is added to the document it also increases the similarity score as it is matching the terms in the query. So we can say that the similarity score will again increase in this case as well.

In conclusion, augmentations that include phrases unrelated to the query ("Fire" in (a) and (b)) are likely to have a some negative effect on similarity. In contrast, augmentations that introduce terms directly linked to the question ("Gold" in (c) and (d)) should boost the similarity. The greater the number of relevant terms added, the greater the likelihood of resemblance. Therefore, the documents may be retrieved in this sequence 4,3,1,2.

3 Question 3

In the term weighting schemes covered in class thus far, we have considered the **tf** factor, the **idf** factor and normalisation approaches.
Assuming that your document collection consists of all the scientific articles published in the Communications of the **ACM (www.acm.org/dl)**, identify two other sources of evidence (features or sets of features) one could consider and suggest a weighting scheme that incorporates these features.
Your answer should define the evidence/feature, your reason for including it, and a means to include it in the weighting scheme.
Ans.
When we are considering the similarity of two scientific documents it is preferred to take other features under considerations as well, and not just bag of words. This will help to enhance the similarity score between the documents and more accurate answers can be given to the user.
To explain this we can take ***Abstracts*** and ***References*** to calculate the similarity score.

1. ***Abstract as Feature 1***:-

   Feature Selection- Generally the abstract are well structured and contains key points, contributions, and findings presented in the paper. It provides a summary of the paper's content.

   Reason- The words are properly organized and contain important keywords and concepts that contains the main ideas of the paper.

2. ***References as Feature 2***:-

   Feature Selection- It provides a list of related work papers that the current paper cites. It provides the information of related work in particular field.

   Reason- References help to establish the link between the current document and the cited documents which can help to increase the similarity score.

   Examples

   For Abstract

   Suppose **Paper A** is on "*Machine Learning for Cancer detection*" with the abstract mentioning terms like "*deep learning*", "*neural networks*", and "*image classification*". **Paper B** on "*Natural Language Processing with Recurrent Neural Networks*" with the abstract mentioning terms like "*RNN's*", "*sequence-to-sequence models*", and "*language translation*". The model can recognize that **Paper A** and **Paper B** are both connected to neural networks and it's application, even if the main body of the papers may differ.

   For References

   Suppose **Paper A**, which has the references of papers on *KNN* and *weighted KNN* in its references. Paper B also cites papers on *KNN* and *weighted KNN*. By including terms from the references in the *TF-IDF* calculation, the model will understand that Paper A and **Paper B** share common citation/references, indicating a certain level of similarity.

   Integration of Selected Features:-

   We can use the linear approach to integrate these features into the weighing scheme. Here, we are using the *TF-IDF* weighing scheme which is suitable for calculating the similarity score.

   Aggregated, $\textbf{Score(d)} = w1 \times \textit{TF-IDF}\ \text{Score(d)} + w2 \times \textit{Abstract Score(d)} + w3 \times \textit{References Score(d)}$.

   Here, **w1**, **w2**, and **w3** are the weights of *TF-IDF* score, abstract score, and references score, respectively.

   By adding these features into the weighing scheme the model gains a better understanding of document relevance and similarity, which will give more accurate results. By adding these terms in the *TF-IDF* calculation, we will be boosting the the similarity score which will help the model to understand the relevancy and can give the appropriate results.