Name: Aditi Vilas Sonkusare

Student Id: 22224352

Information Retrieval: Assignment 1

Q1. 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. (10 marks)

Answer1: In Information Retrieval, preprocessing simplifies the understanding of the query. In Preprocessing, the length of the material is shortened to improve the performance of the system. There are various techniques to simplify the format of the given query. Following are the approach and benefits of the techniques:

1. Tokenization: - Tokenization helps to split the query into small tokens. For splitting the queries into tokens different methods are applied. The main advantage of tokenization is to remove the noise causing words, symbols, and the numbers that may affect the accuracy of query. Benefit of the technique is that the tokens helps in learning and also defining the meaning of the text by analyze the order of words. This method can be in two types:

Given document: Yashashree is a good student. She is studious and hardworking.

Type 1: Tokenizing given document in sentence:

→ [“Yashashree is a good student”. “She is studious and hardworking”]

Type 2: Tokenizing given document into separate words:

→[“Yashashree”,“is”,“a”,“good”,“student”,“.”,“She”,“is”,“studious”,“and”,“hardworking”,“.”

2. Lemmatization: - Lemmatization is similar to stemming, i.e. it also converts a complex word into its root form. The minor difference between both is that lemmatization ensures that the word belongs to its language. We get relevant word when we use this technique. For example,

“There are different types of bats which are hanging on the racks.”

→ “There are different type of bat which are hang on the rack”.

Here, “types → type”, “bats → bat”, “hanging →hang”, “racks → rack” i.e. the given words in the document are converted into the root form.

The benefit of this technique is that the lemma algorithm knows that “worse” is derived from the word “bad” and in stemming “worse” root form shown would be “wor” or something similar to it.

3. Removal of Hyphens/Punctuation Marks: - Hyphens/punctuation marks are treated differently. The full sentence with the hyphens/punctuation marks may be used, or they may not be used. Also, in some models, the hyphen/punctuation marks is omitted or even changed with a space. The benefit of the technique is that by removing hyphens/ punctuations we don’t get a similar word repeatedly.

For example, if we don’t use this technique then; “Their’s → Theirs”, “They? → They” are treated separately.

Q2. 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)

Answer2.

	Counts						Weights
Terms	D1	D2	D3	dfi	D/dfi	idfi	w(i) = tfi.idfi
a	1	1	1	3	3/3=1	0	0
arrived	0	1	1	2	3/2=1.5	0.176	0
Damaged	1	0	0	1	3/1=3	0.477	0.477
Delivery	0	1	0	1	3/1=3	0.477	0
Fire	1	0	0	1	3/1=3	0.477	0.477
gold	1	0	1	2	3/2=1.5	0.176	0.176
in	1	1	1	3	3/3=1	0	0
of	1	1	1	3	3/3=1	0	0
silver	0	2	0	1	3/1=3	0.477	0
shipment	1	0	1	2	3/2=1.5	0.176	0.176
truck	0	1	1	2	3/2=1.5	0.176	0

We know that, the inverse document frequency, idfi = log N/ni---(1)

Tf-Idf weighting scheme:

w (i,j) = f(i,j) x log N/ni

where, f(i,j) = freq (i,j)/len(i,j);

N = number of documents in collection;

ni = number of documents that contains term ti;

f(i,j) = normalized frequency of term ti

→ The columns terms, D1, D2, D3 and dfi shows the counts for each word in the given documents

→ The column dfi contains the frequency for each document; D/dfi, idfi = log D/dfi, where D=3 as there are three documents given. Therefore, the calculation is simple.

→ The w(i) column contains the products and the weight for D1.

Conclusion: In the given question, three documents are provided, therefore d=3. From the equation(1), when the term occurred is only 1 out of the 3 documents, then, its idf = log d/dfj → log 3/1 = 0.477

when the term occurred is only 2 out of the 3 documents, then, its idf = log d/dfj → log 3/2 = 0.176

when the term occurred is only 3 out of the 3 documents, then, its idf = log d/dfj → log 3/3 = 0

Q3. 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.

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

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

➔(A) Answer:

Inverted Indexing Technique: -

In the inverted matrix, we count all tokenized terms and similar terms also: -

Doc ID	Documents	Tokenized term
1	Yashu likes dancing	"Yashu" "likes" "dancing"
2	Yashu likes dancing and playing	"Yashu" "likes" "dancing" "and" "playing"
3	Aditi and Yashu likes dancing	"Aditi" "and" "Yashu" "likes" "dancing"

Sparse Matrix: -

Matrix			
	D1	D2	D3
Yashu	1	1	0
likes	1	1	1
dancing	1	1	1
and	0	1	0
playing	0	1	0
Aditi	0	0	1

A vector is created where document id is shown, referred the column 1 and then the terms are sorted in alphabetical order and respective document id is given to it from the reference: -

	1	Doc ID		2	Doc ID
→	Yashu	1	→	Aditi	3
	likes	1		and	2
→	dancing	1	→	and	3
	Yashu	2	Terms arranged in alphabetical order	dancing	1
→	likes	2		dancing	2
Tokenized Terms	dancing	2		dancing	3
→	and	2	→	likes	1
	playing	2		likes	2
→	Aditi	3	→	likes	3
	and	3		playing	2
→	Yashu	3		Yashu	1
	likes	3		Yashu	2
	dancing	3		Yashu	3

The above step is necessary for the further step of posting list.

Below the table is showing Inverted Index Structure, it is grouped with reference to respective terms and then array structure is created which contains the document id of the respective group: -

INVERTED INDEX				
Term Document	Document Frequency	Posting Lists		
Aditi	1	3		
and	2	2	3	
dancing	3	1	2	3
likes	3	1	2	3
playing	1	2		
Yashu	3	1	2	3

Conclusion: -

Inverted Matrix is very similar to term document matrix. The Inverted Matrix is only storing documents where a respective term appears.

➔**(B) Answer:**

Step 1: Initialized an array score [] = 0 which will store the scores for each document.

Step 2: Initialized an array length which will store the length i.e. normalization factors for each N document.

Step 3 and 4: For each term t in query, calculate weight of term in query, fetch the posting list for the term.

Step 5: For each pair i.e. document-term frequency for a document in a posting list.

Step 6: Here we are calculating the weight in the query vector for term t.

Step 7 and 8: Here we calculate the dot product of the weight frequency of the term in document with the weight of terms in the query array of scores. The scores are updated in the document by adding the contribution from the term.

Step 9: Cosine Similarity is calculated using the similarity of the document i.e. the dot product of the documents divided by the lengths of the documents.

Step 10: Lastly we have to return the top K components with the maximum scores.

Q3 (B) https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf Referred from the reference book.