Sheetal Parikh
Problem Set – Module 4
605.744.81 Information Retrieval
Fall 2021

1.) What do Salton and Buckley mean by tfc term weighting? What is the difference between tfc and nfc term weighting?

TFC term weighting is a method of term weighting described by Salton and Buckley in which you use a normalized tf x idf weight for document terms. Six term-weighting experiments are described in the article in which Salton and Buckley used different combinations of term frequency, collection frequency and length normalization of numerous datasets of different sizes. These combinations were described by two triples representing the term-weighting components used for the document terms (first triple) and the query terms(second triple). They wanted to determine which combination of term-weighting components enhanced retrieval effectiveness so that the items that are relevant to the user was retrieved while those items that are unnecessary were rejected.

The combination "tfc" represented the best overall combination for document weighting. The "t" is a term frequency component and represents regular term frequency, meaning the number of times a word occurs in a document. The "f" is a collection frequency component and represents the idf weight, meaning the inverse document frequency, log(N/n) in which N is the number of documents in the collection and n is the number of documents in which the word appears. Lasty, the "c" is a normalization component for equalizing the length of the document vectors , and represents that a normalization component should be used ("x" would mean that the we shouldn't use any normalization). Generally, when a large number of terms is used, there is a higher chance of terms matching between the queries and documents. Therefore, a normalization factor can be incorporated in order to ensure that all documents, regardless of size, are treated as equally important for retrieval purposes. Salton and Buckley use cosine normalization where each term weight is divided by a factor representing Euclidean vector length.

Another method, "nfc" also performed very well for document weighting. As explained above the "f" and "c" represented using idf for the collection frequency component and that a normalization factor should be used to equalize the length of the document vectors, respectively. However, the "n" represents using an augmented normalized term frequency component in which the term frequency is normalized by the maximum term frequency in the document vectors and lies between 0.5 and 1. Using the normalized term frequency "n" as a term-weighting component is preferable for when retrieving information from collections that contain technical and meaningful terms such as CRAN and MED collections. However, using "t", the regular term frequency, is preferable for when retrieving information from collections where we have more varied and general vocabulary.

2.) The Porter stemmer conflates the words program, programs, and programming into the same stem (program). For a given document collection, how would both document term frequency (TF) and inverse document frequency (IDF) weights change in an IR system using the Porter stemmer compared to an IR system that uses plain words as indexing terms? Provide examples with your explanation.

In an IR system that uses the Porter Stemmer, the term frequency would increase compared to IR systems using plain words causing the inverse document frequency weights to change depending on how common the word becomes in the corpus due to stemming. The words that appear frequently in

the corpus would have lower idf weights due to becoming more common amongst all the documents. However, idf weights may also remain the same for the less common words in the corpus.

The porter stemmer would cause many of the document term frequencies to increase because the count for many words that may have been considered different. For example, after stemming both program and programs would be considered the same word, program, and so the counts for program and programs would be combined. The idf would change depending on the increase/decrease in the occurrence of the word in the corpus. For example, if a word may not be greatly impacted by stemming, the idf would remain the same. However, if the occurrence of a common word increases, the idf may decrease. The change in the combined tf-idf weights would therefore also increase/decrease depending on whether the word is a rare word or a common word in the corpus.

Below is an example of how stemming may impact the term frequency and inverse document frequency:

No Stemming – Using Plain words

	D1 - tf	D2 - tf	D3 - tf	**DF**	**IDF**
apple	1	1	1	3	$\log_2(3/3) = 0$
apples	1	1	0	2	$\log_2(3/2) = 0.5850$
banana	1	0	0	1	$\log_2(3/1) = 1.5850$
bananas	1	0	0	1	$\log_2(3/1) = 1.5850$
kiwi	1	1	0	2	$\log_2(3/2) = 0.5850$
kiwis	0	1	0	1	$\log_2(3/1) = 1.5850$
orange	1	0	2	2	$\log_2(3/2) = 0.5850$
oranges	1	1	1	3	$\log_2(3/3) = 0$

The three document rows(rows 2-3) lists the term frequency of each word in the document. The DF row, lists the number of documents in which each word occurs and the IDF row lists the idf weight for each word.

TF – IDF

	D1	D2	D3
apple	0	0	0
apples	0.5850	0.5850	0
banana	1.5850	0	0
bananas	1.5850	0	0
kiwi	0.5850	0.5850	0
kiwis	0	1.5850	0
orange	0.5850	0	1.17
oranges	0	0	0

The TF-IDF is calculated by multiplying the term frequencies of each document for each word by the idf for each word.

Using Porter Stemmer

	D1 - tf	D2 - tf	D3 - tf	**DF**	**IDF**
apple	2	2	1	3	$\log_2(3/3) = 0$

banana	2	0	0	1	$\log_2(3/1) = 1.5850$
kiwi	1	2	0	2	$\log_2(3/2) = 0.5850$
orange	2	1	3	3	$\log_2(3/3) = 0$

The three document rows(rows 2-3) lists the term frequency of each word in the document. The DF row, lists the number of documents in which each word occurs and the IDF row lists the idf weight for each word.

TF – IDF

	D1	D2	D3
apple	0	0	0
banana	3.17	0	0
kiwi	0.5850	1.17	0
orange	0	0	0

The TF-IDF is calculated by multiplying the term frequencies of each document for each word by the idf for each word.

As can be seen in the example above, the term frequencies increase for most words since the counts of the base word and plural word are combined. The terms apple and orange are common in the corpus so the idf remained 0 or reduced to 0 from when using plain text to the porter stemmer. The term banana was a rare term before and still is after using the stemmer so the idf remained the same, whereas the idf for kiwi reduced a little as it's count increased. This similarly impacted the tf-idf weights. For banana, since the idf remained the same but the tf increased after using the porter stemmer, the tf-idf weight for banana increased from 1.5850 to 3.17. The tf-idf weight for kiwi either remained the same(0.5850) or reduced slightly in the document(1.5859 to 1.17). The tf-idf weights reduced to 0 for all documents for orange and apple as their counts increased after using the porter stemmer.

3.) Cluster Pruning is a method described in Chapter 7 of IIR to restrict search to a subset of a collection based on randomly assigned clusters. Explain what the two parameters b1 and b2 are used for. Give a brief argument either for or against (your choice) for setting both b1 and b2 > 1.

In cluster pruning we randomly chose $\sqrt{N}$ documents to be leaders. All the remaining documents fall into $\sqrt{N}$ groups that are designated as the followers. Cosine scores are computed between the followers and leaders in order to pick the leaders that are the closest to the clusters of followers. Each leader will have approximately $\sqrt{N}$ followers as the leaders are distributed randomly in the document space. Then during query processing, we will want to choose the leader that is the closest to the query.

Cluster pruning however, can be accomplished through different methods using the parameters b1 and b2. In the preprocessing step, we can change the parameter b1 to specify the number of leaders that followers can have. For example, if b1 = 3, then we are choosing 3 leaders for each cluster of followers. These leaders would be the closest 3 leaders to the followers. At query time, we can also change the parameter b2 to specify the number of how many leaders we would want a query to have. For example, if we select b2 = 4, then a query can have 4 leaders. Varying b1 and b2 overall allows us to examine a higher number of documents.

In general, I would argue against setting both b1 and b2 > 1 because although we would be examining a much larger number of documents by increasing b1 and b2, thereby increasing the recall, I believe this would also however, increase the amount of non-relevant information returned, decreasing the precision. Furthermore, searching through a larger number of documents would also increase the cost and time for processing. Therefore, for general searches I believe it is sufficient to keep b1 and b2 at

one. The parameters for b1 and b2 should only be changed for when searching through specific documents for which we know it is necessary to cast a larger retrieval net.

*4.) Calculate cosine similarities for query Q against documents D1 and D2 from the following term-document matrix using the vector cosine model with TF/IDF weighting. Query Q contains the four words **asiago brie brie derby**. The numbers in the table below are term frequencies. The document collection consists of only these eight documents, and the five terms listed below are the only ones found in any document. Recall that for the TF/IDF scheme, the weights for terms in a query or in a document should be the term frequency times the inverse document frequency for that term. Compute accurate cosine scores that do consider the vector length of the query. Show the work in your calculations.*

	D1	D2	D3	D4	D5	D6	D7	D8	**Query: TF**		**DF- words**	**IDF- words**
asiago					4				1		1	$log_2 \frac{8}{1} = log_2 8 = 3$
brie	1	2				1		2	2		4	$log_2 8/4 = log_2 2 = 1$
cheddar	2	2							0		2	$log_2 \frac{8}{2} = log_2 4 = 2$
derby	2	3	3	2	3	3	2	2	1		8	$log_2 8/8 = log_2 1 = 0$
edam	3		4	4			5		0		4	$log_2 \frac{8}{4} = log_2 2 = 1$

TF x IDF

	D1	D2	Query
asiago	$0(3) = \mathbf{0}$	$0(3) = \mathbf{0}$	$1(3) = \mathbf{3}$
brie	$1(1) = \mathbf{1}$	$2(1) = \mathbf{2}$	$2(1) = \mathbf{2}$
cheddar	$2(2) = \mathbf{4}$	$2(2) = \mathbf{4}$	$0(2) = \mathbf{0}$
derby	$2(0) = \mathbf{0}$	$3(0) = \mathbf{0}$	$1(0) = \mathbf{0}$
edam	$3(1) = \mathbf{3}$	$0(1) = \mathbf{0}$	$0(1) = \mathbf{0}$
Sum of Squares	$1^2 + 4^2 + 3^2 = \mathbf{26}$	$2^2 + 4^2 = \mathbf{20}$	$3^2 + 2^2 = \mathbf{13}$
Length	$\sqrt{26} = \mathbf{5.0990}$	$\sqrt{20} = \mathbf{4.4721}$	$\sqrt{13} = \mathbf{3.6056}$
Dot Product	$1(2) + 4(0) + 3(0) = \mathbf{2}$	$2(2) + 4(0) = \mathbf{4}$	
Cosine Score	$\frac{2}{(5.0990)(3.6056)} = \mathbf{0.1088}$	$\frac{4}{(4.4721)(3.6056)} = \mathbf{0.2481}$	

The cosine score for the query against D1: 0.1088
The cosine score for the query against D2: 0.2481