

Input:

- 1) Set of documents (corpa)
- 2) Query
- 3) Relevance Judgment(how relevant document is with the query)

- 1) Tokenization: A document is treated as a string (or bag of words), and then partitioned into a list of tokens.
- 2) Removing stop words: Stop words are frequently occurring, insignificant words. This step eliminates the stop words.
- 3) Stemming word: This step is the process of conflating tokens to their root form (connection -> connect).
- 4) Generation of N-distinct words from the corpora and call them as index terms (or the vocabulary). The document collection is then represented as a N-dimensional vector in term space.
- 5) Computation of Term weights
 - a. Term Frequency.
 - b. Inverse Document Frequency.
 - c. Compute the TF-IDF weighting.

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$											
Query, Q: "gold silver truck"											
D ₁ : "Shipment of gold damaged in a fire"											
D ₂ : "Delivery of silver arrived in a silver truck"											
D ₃ : "Shipment of gold arrived in a truck"											
D = 3; IDF = log(D/df _i)											
Terms	Counts, tf _i				df _i	D/df _i	IDF _i	Weights, w _i = tf _i *IDF _i			
	Q	D ₁	D ₂	D ₃				Q	D ₁	D ₂	D ₃
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

d. Buckley Weight

$$a_{ij} = \frac{\left(0.5 + 0.5 \frac{tf_{ij}}{\max tf}\right) \times \log \frac{N}{n_i}}{\sqrt{\left(0.5 + 0.5 \frac{tf_{ij}}{\max tf}\right)^2 \times \left(\log \frac{N}{n_i}\right)^2}}$$

where a_{ij} is the weight assigned to the term t_j in document D_i , tf_{ij} is the number of times that term t_j appears in document D_i , n_j is the number of documents indexed by the term t_j and finally, N is the total number of documents in the database.

e. Singhal weight

$$\sum_{t \in Q, D} \frac{1 + \ln(1 + \ln(tf))}{(1 - s) + s \frac{df}{avdl}} \times qtf \times \ln \frac{N + 1}{df}$$

Where s is an empirical parameter (usually 0.20), and

- 6) Measuring similarity between document and query.[Following measures has to be used]
- cosine similarity** -The cosine similarity is calculated by measuring the cosine of the angle between document and query.
 - Inner (formula given below)**
 - Dice (formula given below)**
 - Jaccard (formula given below)**
 - Overlap (formula given below)**
 - Assymmetric (formula given below)**
 - Okapi**

NAME	FORMULA
Inner Product	$Sim(D_i, Q_j) = \sum_{x=1}^m (w_{ix} \cdot w_{jx})$
Cosine	$Sim(D_i, Q_j) = \frac{\sum_{x=1}^m (w_{ix} \cdot w_{jx})}{\sqrt{\sum_{x=1}^m (w_{ix})^2 \cdot \sum_{x=1}^m (w_{jx})^2}}$
Dice	$Sim(D_i, Q_j) = \frac{2 \cdot \sum_{x=1}^m (w_{ix} \cdot w_{jx})}{\sum_{x=1}^m (w_{ix})^2 + \sum_{x=1}^m (w_{jx})^2}$
Jaccard	$Sim(D_i, Q_j) = \frac{\sum_{x=1}^m (w_{ix} \cdot w_{jx})}{\sum_{x=1}^m (w_{ix})^2 + \sum_{x=1}^m (w_{jx})^2 - \sum_{x=1}^m (w_{ix} \cdot w_{jx})}$

$$SIM_4(DOC_i, DOC_j) = \frac{\sum_{k=1}^{\dagger} (TERM_{jk} \cdot TERM_{ik})}{\min\left(\sum_{k=1}^{\dagger} (TERM_{jk})^2, \sum_{k=1}^{\dagger} (TERM_{ik})^2\right)} \quad (4)$$

Overlap measure =

$$SIM_5(DOC_i, DOC_j) = \frac{\min\sum_{k=1}^{\dagger} (TERM_{jk}, TERM_{ik})}{\sum_{k=1}^{\dagger} TERM_{jk}} \quad (5)$$

Asymmetric measure =

(SIM 4 is overlap measure and SIM 5 is Assymmetric measure ,here instead of both document ,one is document and other is query)

Okapi =

tf = the term's frequency in document

qtf = the term's frequency in query

N = the total number of documents in the collection

df = the number of documents that contain the term

dl = the document length (in bytes), and

avdl = the average document length

Okapi weighting based document score:

$$\sum_{i=Q,D} \ln \frac{N-df+0.5}{df+0.5} \cdot \frac{(k_1+1)tf}{(k_1((1-b)+b\frac{dl}{avdl})) + tf} \cdot \frac{(k_3+1)qtf}{k_3+qtf}$$

k_1 (between 1.0-2.0), b (usually 0.75), and k_3 (between 0-1000) are constants.

Finally we sort and rank the documents in descending order according to the similarity values

Rank 1: Doc 2 = 0.8246

Rank 2: Doc 3 = 0.3271

Rank 3: Doc 1 = 0.0801

Calculate Recall and precision

i. PRECISION

Given a rank-ordered vector V of results $\langle v_1, \dots, v_n \rangle$ to query q , let $rel(v_i)$ be 1 iff v_i is relevant to q and 0 otherwise. The **precision** of V at document cut-off value k is the number of relevant documents in the top k results:

$$P@k(V) = \frac{1}{k} \sum_{i=1}^k rel(v_i)$$

ii. Mean Average Precision

Given a rank-ordered vector V of results $\langle v_1, \dots, v_n \rangle$ to query q , the **average precision** of V at document cut-off value k is the mean of the precisions at every relevant document (or 0 if there are none):

$$AP@k(V) = \text{avg}_{v_i: i \leq k \wedge rel(v_i)=1} P@i(V) = \frac{\sum_{i=1}^k P@i(V) rel(v_i)}{\sum_{i=1}^k rel(v_i)}$$

The mean average precision of the test set is the mean of the AP's of the queries in the test set.

iii. Mean Reciprocal Rank (MRR)

Given a rank-ordered vector V of results $\langle v_1, \dots, v_n \rangle$ to query q , the **reciprocal rank** of V at document cutoff value k is:

$$RR@k(V) = \begin{cases} \frac{1}{i} & \text{if } \exists i < k : rel(v_i) = 1 \wedge \forall j < i : rel(v_j) = 0 \\ 0 & \text{otherwise} \end{cases}$$

The mean reciprocal rank of the test set is the mean of the RR's of the queries in the test set.

iv. Normalized Discounted Cumulative Gain (NDCG)

Given a rank-ordered vector V of results $\langle v_1, \dots, v_n \rangle$ to query q , let $label(v_i)$ be the judgment of v_i (0=worst, 5=best). The **discounted cumulative gain** of V at document cut-off value k is:

$$DCG@k = \sum_{i=1}^k \frac{1}{\log_2(1+i)} (2^{label(v_i)} - 1)$$

The **normalized DCG** of V is the DCG of V divided by the DCG of the “ideal” (DCG-maximizing) permutation of V (or 1 if the ideal DCG is 0). The NDCG of the test set is the mean of the NDCG’s of the queries in the test set.

v. **F Measure**

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

beta = 1 and alpha 1/2

vi. **Recall**

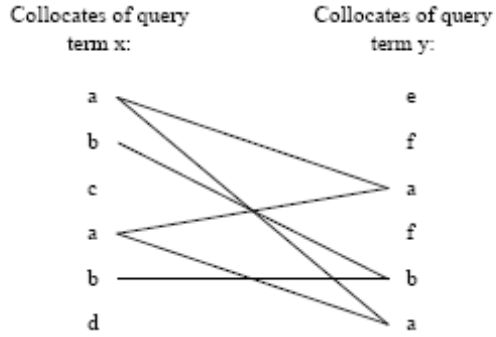
This should be a separate module

Input: [Set of top ranked documents, Query]

- 7) Extract all link terms (words forming links by simple repetition of any two distinct query terms) from each document.

METHOD TO FIND LINK TERMS

Method takes into account how many instance of common collocates each query term has. In Figure, the first column contains collocates in the merged window of the query term x , the second column contains collocates in the merged window of the query term y . The lines between instances of the common collocate in the figure represent lexical links.



In the figure there are altogether 6 links. If there are more than 2 query terms in a document, a comparison of each pair is done. The number of links are recorded for each pair, and summed up to find the total number of links in the document.

To determine similarity of query terms combine all windows for one query term, building a merged window for it. Each query term's merged .In more detail, the algorithm for building merged windows for a query term is as follows:

Fixed-size windows are identified around every instance of a query term in a document. A window is defined as n number of stemmed non-stopwords to the left and right of the query term. We refer to all stemmed non-stopwords extracted from each window surrounding a query term as its *collocates*.

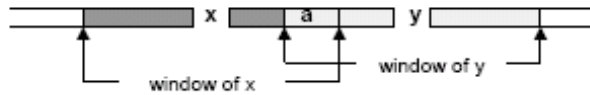


Figure 1: Overlapping windows around query terms x and y.

There could be a situation where windows of two different query terms overlap. In such a case, we run into the following problem: let us assume that query terms x and y have overlapping windows and, hence, both are considered to collocate with term a (see Figure 1). We could simply add this instance of the term a into the merged windows of both x and y .

Doc 1	Doc 2
List of link terms	List of link terms
(Save in either Excel or .txt file)	
Preferably excel	

8) Rank all link terms using idf

FORMULA TO FIND IDF

Assume there are N documents in the collection, and that term t_i occurs in n_i of them. Then the weight to be applied to term t_i , is essentially

$$idf(t_i) = \log \frac{N}{n_i}$$

Doc 1	term Rank 1	termwt
	term Rank n	termwt
Doc 2	term Rank 1	termwt

And 2 link terms per documents are selected

Doc 1	[Top link term1(idf wt) , Top link term2(idf wt)]
Doc 2	

9)

For each document A

Find all link-terms in the document

Rank them by idf

For each of the two top-ranked link-terms in the document A

Find all their instances

For each instance

Identify a snippet around the instance that contains 3 non-stopwords before and after it in text.

End For

Rank all snippets for each link-term by the average idf of their constituent non-stopwords.

End For

End For

A snippet was defined as consisting of 3 non-stop words on either side of the link-term.

Link term1 – Snippet

All the output should be save in .txt files

Rank - Snippets