23/April/2024

CSCE 5200 - Information Retrieval and Web Search

Assignment-6

Uday Bhaskar Valapadasu
11696364

Ans:-

Tables of tf values

Term Frequencies	Doc 1	Doc 2	Doc 3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

Figure 6.9

TERM	df_t	idf_t
car	18,165	1.65
auto	6,723	2.08
insurance	19,241	1.62
best	25,235	1.5

Figure 6.8

terms	Doc1	Doc2	Doc3
car	44.55	6.6	39.6
auto	6.24	68.64	0
insurance	0	53.46	46.98
best	21	0	25.5

tf-idf computed weights.

Computing the euclidean normalized document vectors for each of the documents using the tf-idf weights for terms.?

1) Compute the euclidean length (magnitude) of each document vector using the formula:

$$\boxed{\text{Magnitude}(d) = \sqrt{\sum_{i=1}^{M} (tfidf_i)^2}}$$

2) Normalize each document vector by dividing each tf-idf weight by the document's magnitude.

1) First, compute the Euclidean length (magnitude) of each

→ For Doc 1:

$$\text{Magnitude } (d_{doc1}) = \sqrt{\sum_{i=1}^{M} (tfidf_i)^2}$$

$$= \sqrt{\sum_{i=1}^{4} (tfidf_i)^2}$$

$$= \sqrt{(tfidf_1)^2 + (tfidf_2)^2 + (tfidf_3)^2 + (tfidf_4)^2}$$

$$= \sqrt{44.55^2 + 6.24^2 + 0^2 + 21^2}$$

$$= \sqrt{2464.6401}$$

$$= 49.645141$$

→ For Doc 2:

$$\text{Magnitude } (d_{doc2}) = \sqrt{6.6^2 + 68.64^2 + 53.46^2 + 0^2}$$

$$= \sqrt{7612.9812}$$

$$= 87.252399$$

For Doc 3:

$$\text{Magnitude}(d_{doc3}) = \sqrt{39.6^2 + 0^2 + 46.98^2 + 25.5^2}$$

$$= \sqrt{4425.5304}$$

$$= 66.52466$$

2.) Then, normalize the tf-idf weights for each term in each document by dividing by the magnitude.

For Doc 1:

$$\text{Normalized 'car'} = \frac{44.55}{\text{Magnitude}(d_{doc1}) \text{ of Doc 1}}$$

$$= \frac{44.55}{49.645141}$$

$$= 0.897$$

$$\text{Normalized 'auto'} = \frac{6.24}{49.645141} = 0.125$$

$$\text{Normalized 'insurance'} = \frac{0}{49.645141} = 0$$

Normalized 'best' = $\frac{21}{49.645141} = 0.423$

For Doc2:

Normalized 'car' = $\frac{6.6}{\text{Magnitude}(d_{doc2}) \text{ of doc2}}$

$= \frac{6.6}{87.252} = 0.076$

Normalized 'auto' = $\frac{68.64}{87.252} = 0.786$

Normalized 'insurance' = $\frac{53.46}{87.252} = 0.613$

Normalized 'best' = $\frac{0}{87.252} = 0$

For Doc 3:

Normalized 'car' = $\frac{39.6}{\text{Magnitude}(d_{doc3}) \text{ of doc3}} = \frac{39.6}{66.524}$

$= 0.595$

Normalized 'auto' = $\frac{0}{66.524} = 0$

Normalized 'insurance' = $\frac{46.98}{66.524} = 0.706$

Normalized 'best' = $\frac{25.5}{66.524} = 0.383$

Final Solution Table (Term Weights)

	Doc1	Doc2	Doc3
car	0.897	0.076	0.595
auto	0.125	0.786	0
insurance	0	0.613	0.706
best	0.423	0	0.383

2 Ans:-

1.> For binary terms weights, the formula for the score attribute Score(q,d) for a query q and a document d is:

$$\boxed{Score(q,d) = \sum_{t \in q} (Weight(t,q) \times tf\text{-}idf(t,d))}$$

where weight (t, q) is 1 if the term t is present in the query q, and 0 otherwise

Term	Query		Product		
	tf	W(t,q)	Doc1	Doc2	Doc3
car	1	1	0.897	0.0796	0.595
auto	0	0	0	0	0
insurance	1	1	0	0.613	0.706
best	0	0	0	0	0

→ Score(q, doc1) = 0.897+0+0 = 0.897

→ Score(q, doc2) = 0.0796+0+0.613 = 0.689

→ Score(q, doc3) = ~~0.595+0+0 = 0.595~~
= 0.595+0+0.706 = 1.301

~~Ranking : doc1 > doc2 > doc3~~

Ranking = d3, d1, d2 (doc3 > doc1 > doc2)

2.)

Give the tf-idf weights for "car" and "insurance" in each document (Doc1, Doc2, Doc3), we have:

→ Doc1: tf-idf for "car" = 0.897, tf-idf for "insurance" = 0

→ Doc2: tf-idf for "car" = 0.076, tf-idf for "insurance" = 0.638

→ Doc3: tf-idf for "car" = 0.595, tf-idf for "insurance" = 0.706

The scores given are

→ Doc1: 0.6883

→ Doc2: 0.7975

→ Doc3: 0.7823

ranking

Doc2 > Doc3 > Doc1

3 Ans:

If jealousy and jealous are stemmed to a common term t, then their tf's and their df's would be added together, in computing the tf-idf weights.

Term Frequency (tf):

→ Definition: Term frequency measures how frequently a term occurs in a document. In the context of stemming, the frequency of all terms that are reduced to the same stem are aggregated.

→ Modified Calculation: When "jealous" and "jealousy" are stemmed to "jealous", the tf for "jealous" in a document is the sum of the occurrences of "jealous" and "jealousy" within that document. This approach ensures that the analysis captures all instances of the term, regardless of its specific form.

$tf_{jealous,d}$ = frequency of "jealous" in d + frequency of "jealousy" in d

This calculation provides a more ~~complex~~ comprehensive measure of the term's importance in the document by considering all of its morphological variations.

Inverse Document Frequency (idf)

measures a term's rarity across a document set. For the stem "jealous" idf is calculated considering all variants (like "jealousy") that reduce to this stem. The formula is:

$$idf_{jealous} = \log\left(\frac{N}{df_{jealous}}\right)$$

Here, N is the total number of documents, and $df_{jealous}$ is the count of documents containing "jealous" or "jealousy". This adjustment reflects the term's overall significance across the corpus, pooling all variants under one stem to capture semantic similarities more effectively.

4 Ans:-

To compute the vector space similarity between the query "digital cameras" and the document "digital cameras and video cameras", we need ~~take~~ the following steps:

1.) Calculate term frequency for the query and document: Count the number of times each term appears.

2.) Compute the weighted frequency (wf) for the query: Since logarithmic term weighting is mentioned, use $wf = 1 + \log(tf)$ if $tf > 0$, otherwise $wf = 0$

3.) Calculate inverse document frequency (idf) for the query: use $idf = \log(N/df)$, where N is the total number of documents (10,000,000), and df is the document frequency provided.

4.) Compute q_i for the query: It's stated that $q_i = wf - idf$, so subtract idf from the weighted frequency for the query

5.) Normalize the weighted frequency (wf) for the document vector, so that its length (or magnitude) is 1. This can be done by dividing each wf by the magnitude of the document's vector.

6.) Compute the dot product of q_i and d_i (normalized wf for the document).

7.) The sum of the dot products across all terms will give the cosine similarity score.

word	query					document			
	tf	wf	df	idf	$q_i = wf \cdot idf$	tf	wf	d_i = normalized wf	$q_i \cdot d_i$
digital	1	1	10,000	3	3	1	1	0.52	1.56
video	0	0	100,000	2	0	1	1	0.52	0
cameras	1	1	50,000	2.3	2.3	2	1.3	0.68	1.56

Similarity score : $1.56 + 1.56 = 3.12$

Normalized similarity score is also correct: $3.12 / \text{length(query)}$

$= 3.12 / 3.78$

$= 0.825$

5 Ans:-

Omit this term from the query and proceed; its contribution to the dot product with any document will be zero.

Given this situation, the approach detailed in the solution is to simply omit the non-indexed query term from the calculations:

1.) Omission of Non-indexed Term:

The non-indexed query term is ignored. It is as if this term was never part of the query because it cannot contribute to the similarity score.

2.) Dot-Product Calculation:

The similarity score between a query and a document is typically the dot product of their vectors; without the non-indexed term, the calculation proceeds with the remaining terms.

3.) Vector Space Model (VSM): Represents queries and documents as vectors of terms in a multi-dimensional space

4.) Non-indexed Query Terms: Sometimes a query might contain a term that is not a part of the indexed terms from the document collection, leading to no matching dimension in the VSM.

5.) Zero-contribution principle: A term not present in the document collection contributes zero to the dot product (similarity score) since it has no associated weight in the documents vectors.

6.) Similarity Score calculation: The final similarity score excludes the non-indexed term, relying only on the contributions from indexed terms.

7.) Potential Drawbacks: This approach assumes the missing term is not critical to the query's meaning. If the term is significant,

its omission could potentially affect the relevancy of the search results.

8.) <u>Trade-off</u>; There is a balance between computational efficiency and the fidelity of query representation. Maintaining the integrity of the vector space comes at the risk of neglecting potentially meaningful query terms.

∴ Omitting non-indexed query terms ~~sim~~ simplifies VSM computations but may miss key aspects of the user's search intent.