**Implementation of cosine similarity score using an example (Concept understood from https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/ and Croft textbook)**

1)Find the term frequency(tf) of each term in the document

2)Find the normalised term frequency of the terms which is equal to tf divided by no. of terms in the document

For e.g.: consider this document:

**Document 1**: The game of life is a game of everlasting learning

**TF for Document 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Document1** | **the** | **game** | **of** | **life** | **is** | **a** | **everlasting** | **learning** |
| **Term Frequency** | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |

**Normalized TF for Document 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Document1** | **the** | **game** | **of** | **life** | **is** | **a** | **everlasting** | **learning** |
| **Normalized TF** | 0.1 | 0.2 | 0.2 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |

In reality each document will be of different size. On a large document the frequency of the terms will be much higher than the smaller ones. Hence we need to **normalize** the document based on its size. Otherwise the documents of larger length will be having more score which is not always right.

3)Inverse document frequency:

The main purpose of doing a search is to find out **relevant** **documents** matching the query. In the first step all terms are considered equally important. In fact, certain terms that occur too frequently have little power in determining the relevance. We need a way to **weigh down** the effects of too frequently occurring terms. Also the terms that occur less in the document can be more relevant. We need a way to **weigh up** the effects of less frequently occurring terms.

Idf(term) = 1.0 + log(float(len(allDocuments)) / numDocumentsWithThisTerm)

4) Weight of a term=tf\*idf

In this step we are deriving vectors for documents. The vector for a document is obtained by taking the product of tf\*idf for each unique term in the document. The weights are obtained for all the documents.

5) The same process is repeated for query. For query, the normalised term frequency is the frequency of that term in the query divided by the total number of terms in the query. And the inverse document frequency is 1.

6)

Cosine Similarity (Query, Document1) = Dot product (Query, Document1) / ||Query|| \* ||Document1||

Here we are calculating the score for each document based on the query.

Dot product (Query, Document1) is the sum of product of weight of a term in the document and its weight in the query for all the terms.

||Document1|| is the magnitude of document vector

||Query|| is the magnitude of query vector

Once score for all the documents are obtained we sort them in the decreasing order of their ranks and the top 100 results are filtered out.