**Concept understood from** [**https://lucene.apache.org/core/3\_6\_0/api/core/org/apache/lucene/search/Similarity.html**](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/Similarity.html)

**The differences in output for Lucene and our implementation is mainly due to:**

* **Difference in tokenising**
* **Difference in scoring (Weight calculation in particular) used in Lucene and our implementation.**

**These are explained in detail below:**

* **Scoring**

**SCORING FUNCTION USED BY LUCENE**

In case of Lucene, we are inputting the corpus obtained from the HW3. The parsing and scoring is done using the algorithms already defined in Lucene. For scoring, Lucene combines Boolean model (BM) of Information Retrieval with Vector Space Model (VSM) of Information Retrieval.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | cosine-similarity(q,d)   = | |  | | --- | | V(q) · V(d) | | ––––––––– | | |V(q)| |V(d)| | | | |
| VSM Score |

Where *V(q)* · *V(d)* is the [dot product](http://en.wikipedia.org/wiki/Dot_product) of the weighted vectors, and *|V(q)|* and *|V(d)|* are their [Euclidean norms](http://en.wikipedia.org/wiki/Euclidean_norm#Euclidean_norm).

The weights of term in the vector is calculated as follows:

1. ***tf(t in d)*** correlates to the term's *frequency*, defined as the number of times term *t* appears in the currently scored document *d*. Documents that have more occurrences of a given term receive a higher score. Note that *tf(t in q)* is assumed to be *1* and therefore it does not appear in this equation, However if a query contains twice the same term, there will be two term-queries with that same term and hence the computation would still be correct (although not very efficient). The default computation for *tf(t in d)* in [DefaultSimilarity](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html" \l "tf(float)) is:

|  |  |
| --- | --- |
| [tf(t in d)](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html#tf(float))   = | frequency½ |

1. ***idf(t)*** stands for Inverse Document Frequency. This value correlates to the inverse of *docFreq* (the number of documents in which the term *t* appears). This means rarer terms give higher contribution to the total score. *idf(t)* appears for *t* in both the query and the document, hence it is squared in the equation. The default computation for *idf(t)* in [DefaultSimilarity](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html" \l "idf(int, int)) is:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [idf(t)](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html#idf(int, int))  = | 1 + log ( | |  | | --- | | numDocs | | ––––––––– | | docFreq+1 | | ) |

**3 *queryNorm(q)***is a normalizing factor used to make scores between queries comparable. This factor does not affect document ranking (since all ranked documents are multiplied by the same factor), but rather just attempts to make scores from different queries (or even different indexes) comparable. This is a search time factor computed by the Similarity in effect at search time. The default computation in [DefaultSimilarity](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html" \l "queryNorm(float)) produces a [Euclidean norm](http://en.wikipedia.org/wiki/Euclidean_norm#Euclidean_norm): 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| queryNorm(q)   =   [queryNorm(sumOfSquaredWeights)](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html" \l "queryNorm(float))   = | |  | | --- | | 1 | | –––––––––––––– | | sumOfSquaredWeights½ | |

**SCORING FUNCTION USED IN OUR CODE**

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 eg: 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 |

3)Inverse document frequency:

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

So **the way score is calculated for each document is different for both Lucene and our implementation.**

* **PARSING**

Parsing used by Lucene:

We use **Simple Analyser of Lucene** to tokenize the raw data obtained from HW3. Simple Analyser in Lucene does the tokenising by dividing the text at non-letters. And the obtained output is converted to lower case.

In case of our corpus creation, we are removing punctuation (except hyphen), non-ascii characters and also the punctuations within numbers are retained.

So the **corpus on which scoring function is applied** also varies.