# Tf-Idf and Cosine similarity

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

**Document 2**: The unexamined life is not worth living

**Document 3**: Never stop learning

Let us imagine that you are doing a search on these documents with the following query: **life learning**

## Step 1: Term Frequency (TF)

Term Frequency also known as TF measures the number of times a term (word) occurs in a document. Given below are the terms and their frequency on each of the document.

**TF for Document 1**

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

**TF for Document 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Document2** | **the** | **unexamined** | **life** | **is** | **not** | **worth** | **living** |
| **Term Frequency** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**TF for Document 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document3** | **never** | **stop** | **learning** |
| **Term Frequency** | 1 | 1 | 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. A simple trick is to divide the term frequency by the total number of terms. For example in Document 1 the term **game** occurs **two** times. The total number of terms in the document is **10**. Hence the normalized term frequency is **2 / 10 = 0.2**. Given below are the normalized term frequency for all the documents.

**Normalized TF for Document 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 |

**Normalized TF for Document 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Document2** | **the** | **unexamined** | **life** | **is** | **not** | **worth** | **living** |
| **Normalized TF** | 0.142857 | 0.142857 | 0.142857 | 0.142857 | 0.142857 | 0.142857 | 0.142857 |

**Normalized TF for Document 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document3** | **never** | **stop** | **learning** |
| **Normalized TF** | 0.333333 | 0.333333 | 0.333333 |

## Step 2: Inverse Document Frequency (IDF)

Let us compute IDF for the term **game**

IDF(**game**) = 1 + loge(Total Number Of Documents / Number Of Documents with term **game** in it)

There are 3 documents in all = Document1, Document2, Document3

The term game appears in Document1

IDF(**game**) = 1 + loge(3 / 1)

= 1 + 1.098726209

= 2.098726209

Given below is the IDF for terms occurring in all the documents. Since the terms: **the, life, is, learning** occurs in 2 out of 3 documents they have a lower score compared to the other terms that appear in only one document.

|  |  |
| --- | --- |
| **Terms** | **IDF** |
| The | 1.405507153 |
| Game | 2.098726209 |
| Of | 2.098726209 |
| Life | 1.405507153 |
| Is | 1.405507153 |
| A | 2.098726209 |
| Everlasting | 2.098726209 |
| Learning | 1.405507153 |
| Unexamined | 2.098726209 |
| Not | 2.098726209 |
| Worth | 2.098726209 |
| Living | 2.098726209 |
| Never | 2.098726209 |
| Stop | 2.098726209 |

## Step 3: TF \* IDF

Remember we are trying to find out relevant documents for the query: **life learning**

For each term in the query multiply its normalized term frequency with its IDF on each document. In Document1 for the term **life** the normalized term frequency is 0.1 and its IDF is 1.405507153. Multiplying them together we get **0.140550715** (0.1 \* 1.405507153). Given below is TF \* IDF calculations for **life and learning** in all the documents.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Document1** | **Document2** | **Document3** |
| **Life** | 0.140550715 | 0.200786736 | 0 |
| **Learning** | 0.140550715 | 0 | 0.468502384 |

## Step 4: Vector Space Model – Cosine Similarity

. The set of documents in a collection then is viewed as a set of vectors in a vector space. Each term will have its own axis. Using the formula given below we can find out the similarity between any two documents.

Cosine Similarity (d1, d2) = Dot product(d1, d2) **/** ||d1|| \* ||d2||

Dot product (d1,d2) = d1[0] \* d2[0] + d1[1] \* d2[1] \* … \* d1[n] \* d2[n]

||d1|| = square root(d1[0]2 + d1[1]2 + ... + d1[n]2)

||d2|| = square root(d2[0]2 + d2[1]2 + ... + d2[n]2)

**If you’re rusty on trigonometry, all you need to remember to understand this is that the cosine value is always between –1 and 1: the cosine of a small angle is near 1, and the cosine of a large angle near 180 degrees is close to –1. This is good, because small angles should map to high similarity, near 1, and large angles should map to near –1**.

The query entered by the user can also be represented as a vector. We will calculate the TF\*IDF for the query

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TF** | **IDF** | **TF\*IDF** |
| **Life** | 0.5 | 1.405507153 | 0.702753576 |
| **Learning** | 0.5 | 1.405507153 | 0.702753576 |

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

**Dot product(Query, Document1)**

= ((0.702753576) \* (0.140550715) + (0.702753576)\*(0.140550715))

= 0.197545035151

**||Query||** = sqrt((0.702753576)2 + (0.702753576)2) = 0.993843638185

**||Document1||** = sqrt((0.140550715)2 + (0.140550715)2) = 0.198768727354

**Cosine Similarity(Query, Document)** = 0.197545035151 / (0.993843638185) \* (0.198768727354)

= 0.197545035151 / 0.197545035151

= 1

Given below is the similarity scores for all the documents and the query

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Document1** | **Document2** | **Document3** |
| **Cosine Similarity** | 1 | 0.707106781 | 0.707106781 |

Document1 has the highest score of 1. This is not surprising as it has both the terms life and learning.

|  |  |
| --- | --- |
| **What does tf-idf mean?** | |
| Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.  One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model.  Tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.  **How to Compute:**  Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.   * **TF: Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:   TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document). * **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:   IDF(t) = log\_e(Total number of documents / Number of documents with term t in it).   See below for a simple example.  **Example:**  Consider a document containing 100 words wherein the word *cat* appears 3 times. The term frequency (i.e., tf) for *cat* is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word *cat* appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 \* 4 = 0.12. |  |