Write a program in which

Dataset:

Here I have provide you no news group dataset.

The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. **comp.sys.ibm.pc.hardware / comp.sys.mac.hardware**), while others are highly unrelated (e.g **misc.forsale / soc.religion.christian**). Here is a list of the 20 newsgroups, partitioned (more or less) according to subject matter:

|  |  |  |
| --- | --- | --- |
| comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x | rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey | sci.crypt sci.electronics sci.med sci.space |
| misc.forsale | talk.politics.misc talk.politics.guns talk.politics.mideast | talk.religion.misc alt.atheism soc.religion.christian |

**Data**

The data available here are in .tar.gz bundles. You will need [tar](http://www.gnu.org/software/tar/tar.html) and [gunzip](http://www.gnu.org/software/gzip/gzip.html) to open them. Each subdirectory in the bundle represents a newsgroup; each file in a subdirectory is the text of some newsgroup document that was posted to that newsgroup.

Documents consist of set of word. Form the document create dictionary of words used in all documents.

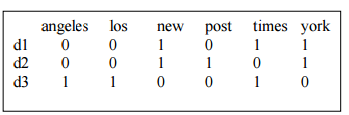
Here is a simplified example of the vector space retrieval model. Consider a very small collection C that consists in the following three documents:

d1: “new york times”

d2: “new york post”

d3: “los angeles times”

in dictionary we have six words(new ,york, times,post,los, angeles). Now every document can be represented as vector of six dimension.



**Term frequency**[[edit](https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&action=edit&section=2" \o "Edit section: Term frequency)]

Suppose we have a set of English text documents and wish to determine which document is most relevant to the query "the brown cow". A simple way to start out is by eliminating documents that do not contain all three words "the", "brown", and "cow", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document and sum them all together; the number of times a term occurs in a document is called its *term frequency*.

The first form of term weighting is due to [Hans Peter Luhn](https://en.wikipedia.org/wiki/Hans_Peter_Luhn) (1957) and is based on the Luhn Assumption:

* The weight of a term that occurs in a document is simply proportional to the term frequency. [[2]](https://en.wikipedia.org/wiki/Tf%E2%80%93idf#cite_note-2)

**Inverse document frequency**[[edit](https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&action=edit&section=3" \o "Edit section: Inverse document frequency)]

However, because the term "the" is so common, this will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "brown" and "cow". The term "the" is not a good keyword to distinguish relevant and non-relevant documents and terms, unlike the less common words "brown" and "cow". Hence an *inverse document frequency* factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

http://www.ir-facility.org/image/108x46ximage_gallery,quuid=483b3f66-3643-40e3-b07b-4e3a6ffec37a,agroupId=10156,at=1299682843095.pagespeed.ic.93XR1MM7jW.png

**Term frequency–Inverse document frequency**[[edit](https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&action=edit&section=7" \o "Edit section: Term frequency–Inverse document frequency)]

* Then tf–idf is calculated as
* \mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t, D)

The total

number of documents is N=3. Therefore, the idf values for the terms are:

angles log2(3/1)=1.584

los log2(3/1)=1.584

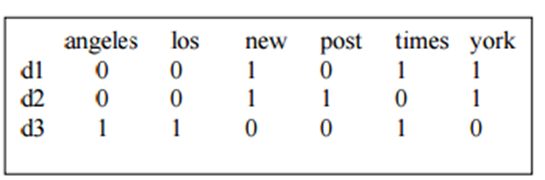
new log2(3/2)=0.584

post log2(3/1)=1.584

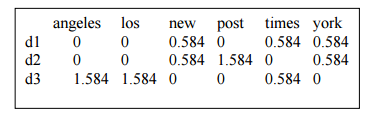
times log2(3/2)=0.584

york log2(3/2)=0.584

For all the documents, we calculate the tf scores for all the terms in C. We assume the words in the vectors are ordered alphabetically.



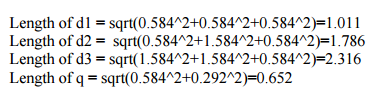
Now we multiply the tf scores by the idf values of each term, obtaining the following matrix of documents-by-terms: (All the terms appeared only once in each document in our small collection, so the maximum value for normalization is 1.)



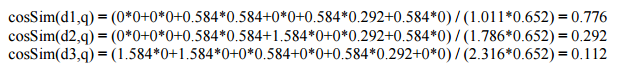
Given the following query: “new new times”, we calculate the tf-idf vector for the query, and compute the score of each document in C relative to this query, using the cosine similarity measure. When computing the tf-idf values for the query terms we divide the frequency by the maximum frequency (2) and multiply with the idf values.



We calculate the length of each document and of the query:



Then the similarity values are:



According to the similarity values, the final order in which the documents are presented as result to the query will be: d1, d2, d3.

### Cosine similarity measure

To avoid the bias caused by different document lengths, a common way to compute the similarity of two documents is using the cosine similarity measure. The inner product of the two vectors (sum of the pairwise multiplied elements) is divided by the product of their vector lengths. This has the effect that the vectors are normalized to unit length and only the angle, more precisely the cosine of the angle, between the vectors accounts for their similarity.

http://www.ir-facility.org/image/208x59ximage_gallery,quuid=ca7a2d08-fe94-4338-86d8-432e4ca57ceb,agroupId=10156,at=1299682843094.pagespeed.ic.E4t9-9gR9L.png

Documents not sharing a single word get assigned a similarity value of zero because of the orthogonality of their vectors while documents sharing a similar vocabulary get higher values (up to one in the case of identical documents). Because a query can be considered a short document, it is of course possible to create a vector for the query, which can then be used to calculate the cosine similarities between the query vector and those of the matching documents. Finally, the similarity values between query and the retrieved documents are used to rank the results.