Dublin City University

School of Computing

CA4009: Search Technologies

Section 3a: Best Match Retrieval Example

Gareth Jones

October 2019

Introduction

This example is intended as an illustration of ranked best-match document retrieval for the classical vector-space and probabilistic retrieval models.

The example takes a simple document collection, indexes it, and shows the calculation of scores for ranked lists using the term weighting mechanisms introduced in lectures.

The intention of the example is to help make it clearer how these information retrieval techniques are used with text documents.

It should be possible to calculate all the numbers at each stage given the information available.

Examples Documents

The example collection consists of the following 5 documents.

D1: Information retrieval is concerned with the location of relevant documents. A search query retrieves documents from a collection.

D2: Searchers often do not know how an information retrieval system works, but search for information anyway.

D3: Managers of search engines often develop their own information retrieval algorithms. These algorithms are often kept secret.

D4: Database systems and information retrieval systems are quite different.

D5: This link is my favourite search engine.

Preprocessing

The documents must be subjected to preprocessing procedures before being placed into a retrieval system.

These preprocessing stages may include stop word removal and suffix stripping.

For this example we will apply both stop word removal and suffix stripping.

In stop word removal any words from a predefined stop word list which appear in the documents will be deleted.

For this example assume that the following words found in the documents are stop words: *is, with, the, of, from, do, not, how, an, but, for, these, are, kept, and, this, my.*

Vocabulary List

The vocabulary of these documents after removing stop words and suffix stripping.

(X) when term frequency in document is greater than 1.

	D1	D2	D3	D4	D5
information	1	1 (2)	1	1	0
retriev	1 (2)	1	1	1	0
concern	1	0	0	0	0
locat	1	0	0	0	0
relev	1	0	0	0	0
documen	1 (2)	0	0	0	0

Vocabulary List

	D1	D2	D3	D4	D5
search	1	1 (2)	1	0	1
query	1	0	0	0	0
collect	1	0	0	0	0
often	0	1	1(2)	0	0
know	0	1	0	0	0
system	0	1	0	1(2)	0
work	0	1	0	0	0
manager	0	0	1	0	0

Vocabulary List

	D1	D2	D3	D4	D5
engine	0	0	1	0	1
develop	0	0	1	0	0
algorithms	0	0	1 (2)	0	0
secret	0	0	1	0	0
database	0	0	0	1	0
quite	0	0	0	1	0
different	0	0	0	1	0
link	0	0	0	0	1
favourite	0	0	0	0	1

Document Vectors

Terms describe the topic of a document. The more frequently a term occurs in a document, (on average) the more this term tells us about the topic of the document.

Examining the sets of terms associated with the documents.

We can see that the presence of 2 occurrences of ”retriev” and ”docum” in D1 indicate that the main focus of the document is document retrieval.

Whereas a group of terms that occur only once could give us a quite false impression of the main topic of the document, for example ”relev” and ”locat” might lead us to conclude that the document is about *relevant locations*.

Document Vectors

Each document is described by a set of representative index terms.

Not all terms are equally useful for describing the contents of a document.

Distinct index terms have varying importance when used to describe document contents.

This effect is captured through the assignment of numerical term weights to each index term in a document.

Let $t(i)$ be an index term in document $d(j)$ and $w(i, j)$ be a weight associated with $t(i)$ in $d(j)$.

Document Vectors

Let V be the number different index terms in the collection.

$V = \{t(1), t(2), \ldots, t(V)\}$ is the set of all index terms.

A weight $w(i, j)$ is associated with each term in each document.

For an index term that does not occur in the document $w(i, j) = 0$.

Each document can be thought of as an index term vector of the form.

$d(j) = (w(i, j), w(2, j), \ldots, w(V, j))$

For the simple case of binary weights $w(i, j) = 0$ or 1 depending on presence or absence of the term in the document.

Document Vectors

Binary vectors for our example documents are as follows:

D1 = $(1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

D2 = $(1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

D3 = $(1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$

D4 = $(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)$

D5 = $(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1)$

Simple Query-Document Matching

Example Query 1: information retrieval algorithms

Q1 = $(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$

Example Query 2: search engine algorithms

Q2 = $(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0)$

Matching score can be computed as the dot product.

$$ms(j) = Q.d(j) = \sum_{i=1}^{V} w(q(i)) \times w(i, j)$$

Note the query terms can have a weighting scheme applied to as well. This is indicated in the general query term weight $w(q(i))$. However, throughout this example we assign query terms the value of 0 or 1.

Simple Query-Document Matching

Matching Score Query 1:

ms(D1) = 2

ms(D2) = 2

ms(D3) = 3

ms(D4) = 2

ms(D5) = 0

Matching Score Query 2:

ms(D1) = 1

ms(D2) = 1

ms(D3) = 3

ms(D4) = 0

ms(D5) = 2

Collection Frequency Weighting

$cfw(i) = idf(i)$

The $cfw(i)$ values for the terms in the example queries can be computed as follows.

$$cfw(i) = \log \frac{N}{n(i)}$$

$cfw(information) = \log \frac{5}{4} = 0.097$

$cfw(retriev) = \log \frac{5}{4} = 0.097$

$cfw(algorithm) = \log \frac{5}{1} = 0.699$

$cfw(search) = \log \frac{5}{4} = 0.097$

$cfw(engine) = \log \frac{5}{2} = 0.397$

$cfw(i)$ Query-Document Matching

Replacing the binary weights with the $cfw(i)$ weight gives:

D1 = $(0.097, 0.097, 0.699, 0.699, 0.699, 0.699, 0.097, 0.699, 0.699,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

D2 = $(0.097, 0.097, 0, 0, 0, 0, 0.097, 0, 0, 0.398, 0.699, 0.398, 0.699,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

D3 = $(0.097, 0.097, 0, 0, 0, 0, 0.097, 0, 0, 0.398, 0, 0, 0, 0.699, 0.398,$
$0.699, 0.699, 0.699, 0, 0, 0, 0, 0)$

D4 = $(0.097, 0.097, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.398, 0, 0, 0, 0,$
$0, 0, 0.699, 0.699, 0.699, 0, 0)$

D5 = $(0, 0, 0, 0, 0, 0, 0.097, 0, 0, 0, 0, 0, 0, 0, 0.398, 0, 0,$
$0, 0, 0, 0, 0.699, 0.699)$

$cfw(i)$ Query-Document Matching

Matching Score Query 1:

ms(D1) = 0.194

ms(D2) = 0.194

ms(D3) = 0.893

ms(D4) = 0.194

ms(D5) = 0.0

Matching Score Query 2:

ms(D1) = 0.097

ms(D2) = 0.097

ms(D3) = 1.193

ms(D4) = 0.0

ms(D5) = 0.494

$tf \times idf$ Query-Document Matching

For the $tf(i,j)$ function $f(tf(i,j)) = \log(x) + 1.$

$tf(i,j) = 0$, $f(tf(i,j)) = 0$ (by definition)

$tf(i,j) = 1$, $f(tf(i,j)) = 1.0$

$tf(i,j) = 2$, $f(tf(i,j)) = 1.301$

$tf(i,j) = 3$, $f(tf(i,j)) = 1.477$

$tf \times idf$ Query-Document Matching

The terms from Query 1 as they appear in D1.

$$w(information, 1) = 0.097 \times (\log(1) + 1) = 0.097$$

$$w(retriev, 1) = 0.097 \times (\log(2) + 1) = 0.126$$

The terms from Query 2 as they appear in D3.

$$w(search, 3) = 0.097 \times (\log(1) + 1) = 0.097$$

$$w(engine, 3) = 0.397 \times (\log(1) + 1) = 0.397$$

$$w(algorithm, 3) = 0.699 \times (\log(2) + 1) = 0.909$$

We can see here that the weight of ”retriev” in D1 is now greater than that of ”information” since it occurs twice in D1.

$tf \times idf$ Query-Document Matching

Replacing the $cfw(i)$ weights with the $tf \times idf$ weights gives:

D1 = $(0.097, \mathbf{0.126}, 0.699, 0.699, 0.699, \mathbf{0.909}, 0.097, 0.699, 0.699,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

D2 = $(0.097, 0.097, 0, 0, 0, 0, \mathbf{0.126}, 0, 0, 0.398, 0.699, 0.398, 0.699,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

D3 = $(0.097, 0.097, 0, 0, 0, 0, 0.097, 0, 0, \mathbf{0.518}, 0, 0, 0, 0.699, 0.398,$
$0.699, \mathbf{0.909}, 0.699, 0, 0, 0, 0, 0)$

D4 = $(0.097, 0.097, 0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{0.517}, 0, 0, 0, 0,$
$0, 0, 0.699, 0.699, 0.699, 0, 0)$

D5 = $(0, 0, 0, 0, 0, 0, 0.097, 0, 0, 0, 0, 0, 0, 0, 0.398, 0, 0,$
$0, 0, 0, 0, 0.699, 0.699)$

$tf \times idf$ Query Document Matching

Matching Score Query 1:

ms(D1) = 0.097 + 0.126 = 0.223

ms(D2) = 0.194

ms(D3) = 0.097 + 0.097 + 0.909 = 1.103

ms(D4) = 0.194

ms(D5) = 0.0

Matching Score Query 2:

ms(D1) = 0.097

ms(D2) = 0.126

ms(D3) = 0.097 + 0.398 + 0.909 = 1.404

ms(D4) = 0.0

ms(D5) = 0.494

Vector-Space Model

The similarity between the query and each document $d(j)$ can be computed as,

$$Sim(q, d(j)) = \frac{\sum_{i=0}^{I-1} w_q(i).w_d(i,j)}{\sqrt{\sum_{i=0}^{I-1} w_q(i)^2}\sqrt{\sum_{i=0}^{I-1} w_d(i,j)^2}}$$

The documents are then ranked in decreasing order of similarity.

The term weights $w(i,j)$ can be any effective weighting scheme, e.g. binary, $cfw(i)$ or $tf \times idf$.

Vector-Space Model

The numerator of this similarity score will be the same as before.

For the denominator we need to compute $\sum_{i=0}^{I-1} w_d(i,j)^2$ for each document, where $w_d(i,j)$ is the weight of i in j which depends on the weighting scheme used.

For $tf \times idf$ weights the modulus of each document is:

D1: $\sqrt{3.30397} = 1.817$

D2: $\sqrt{1.650266} = 1.133$

D3: $\sqrt{2.7495} = 1.658$

D4: $\sqrt{1.755191} = 1.325$

D5: $\sqrt{1.13638} = 1.066$

Vector-Space Model

Also for each query we need to compute $\sum_{i=0}^{I-1} w_q(i)^2$

For the queries this is:

Q1: $\sqrt{3.0} = 1.732$

Q2: $\sqrt{3.0} = 1.732$

Vector-Space Model

Matching Score Query 1:

$$\text{ms(D1)} = 0.223/(1.817 \times 1.732) = 0.0709$$
$$\text{ms(D2)} = 0.194/(1.133 \times 1.732) = 0.0989$$
$$\text{ms(D3)} = 1.103/(1.658 \times 1.732) = 0.3841$$
$$\text{ms(D4)} = 0.194/(1.325 \times 1.732) = 0.0845$$
$$\text{ms(D5)} = 0.0/(1.066 \times 1.732) = 0.0$$

Matching Score Query 2:

$$\text{ms(D1)} = 0.097/(1.817 \times 1.732) = 0.0308$$
$$\text{ms(D2)} = 0.126/(1.133 \times 1.732) = 0.0642$$
$$\text{ms(D3)} = 1.403/(1.655 \times 1.732) = 0.4895$$
$$\text{ms(D4)} = 0.0/(1.325 \times 1.732) = 0.0$$
$$\text{ms(D5)} = 0.494/(1.066 \times 1.732) = 0.2676$$

Probabilistic Model

Okapi BM25 combined weighting $cw(i,j)$ scheme:

$$cw(i,j) = cfw(i) \times \frac{tf(i,j) \times (k_1 + 1)}{k_1 \times ((1-b) + (b \times ndl(j))) + tf(i,j)}$$

where cw indicates the combined weighting scheme, ndl is the normalised length of document j, and k_1 and b are empirically determined constants that control the effect of $tf(i,j)$ and degree of length normalisation respectively.

$$ndl(j) = \frac{dl(j)}{\text{average } dl \text{ for all documents}}$$

Normalized Document Length

The length normalisation element of BM25 depends on the value of parameter b. If $b = 0.0$, there is no length normalization.

In order to use BM25 with a $b \neq 0.0$ we need to compute the normalised document length $ndl(j)$ as follows.

$$avedl = (18 + 16 + 17 + 9 + 7)/5 = 67/5 = 13.4$$

$$dl(1) = 18 \quad ndl(1) = \frac{18}{13.4} = 1.34$$

$$dl(2) = 16 \quad ndl(2) = 1.19$$

$$dl(3) = 17 \quad ndl(3) = 1.27$$

$$dl(4) = 9 \quad ndl(4) = 0.67$$

$$dl(5) = 7 \quad ndl(5) = 0.522$$

Probabilistic Model

For $b = 1$, BM25 can rewritten as,

$$cw(i,j) = cfw(i) \times \frac{tf(i,j) \times (k_1 + 1)}{k_1 \times ndl(j)) + tf(i,j)}$$

Considering the second part of this equation which depends on the value of $tf(i,j)$ and $ndl(j)$. If we assume $k_1 = 1.5$.

Consider the case of a specific document j with average length, i.e. $ndl(j) = 1.0$. For this document, the $tf(i,j)$ component is simply,

$$f(tf(i,j)) = \frac{tf(i,j) \times 2.5}{1.5 + tf(i,j)}$$

Probabilistic Model

For some sample values of $tf(i,j)$.

$tf(i,j) = 0, f(tf(i,j)) = 0.0$

$tf(i,j) = 1, f(tf(i,j)) = \frac{1 \times 2.5}{1.5+1} = 1.0$

$tf(i,j) = 2, f(tf(i,j)) = \frac{2 \times 2.5}{1.5+2} = 1.563$

$tf(i,j) = 3, f(tf(i,j)) = 1.667$

$tf(i,j) = 5, f(tf(i,j)) = 1.923$

$tf(i,j) = 10, f(tf(i,j)) = 2.174$

Probabilistic Model

The terms from Query 1 as they appear in D1.

$$w(information, 1) = 0.097 \times \frac{1 \times 2.5}{1.5 \times 1.34 + 1} = 0.0806$$

$$w(retriev, 1) = 0.097 \times \frac{2 \times 2.5}{1.5 \times 1.34 + 2} = 0.1209$$

The terms from Query 2 as they appear in D3.

$$w(search, 3) = 0.097 \times \frac{1 \times 2.5}{1.5 \times 1.27 + 1} = 0.0835$$

$$w(engine, 3) = 0.397 \times \frac{1 \times 2.5}{1.5 \times 1.27 + 1} = 0.3417$$

$$w(algorithm, 3) = 0.699 \times \frac{2 \times 2.5}{1.5 \times 1.27 + 2} = 0.8950$$

We can see here that the weight of ”retriev” in D1 is again greater than that of ”information” since it occurs twice in D1.

Probabilistic Model

Matching Score Query 1:

ms(D1) $= 0.1209 + 0.0806 = 0.2015$

ms(D2) $= 0.0871 + 0.1281 = 0.2152$

ms(D3) $= 0.0835 + 0.0835 + 0.8950 = 1.0620$

ms(D4) $= 0.1209 + 0.1209 = 0.2419$

ms(D5) $= 0.0 = 0.0$

Matching Score Query 2:

ms(D1) $= 0.0806 = 0.0806$

ms(D2) $= 0.1281 = 0.1281$

ms(D3) $= 0.0835 + 0.3417 + 0.8950 = 1.3202$

ms(D4) $= 0.0 = 0.0$

ms(D5) $= 0.1360 + 0.5566 = 0.6926$

Probabilistic Model

For $b = 0$, BM25 can rewritten as,

$$cw(i,j) = cfw(i) \times \frac{tf(i,j) \times (k_1 + 1)}{k_1 + tf(i,j)}$$

Note: this is exactly the same as the effective version of BM25 with $b = 1$ for a document of average length, i.e. when $ndl(j) = 1.0$.

Probabilistic Model

For the terms from Query 1 as they appear in D1 the $cw(i,j)$ weights when $b = 0$ would now be computed as,

$$w(information, 1) = 0.097 \times \frac{1 \times 2.5}{1.5+1} = 0.0907$$

$$w(retriev, 1) = 0.097 \times \frac{2 \times 2.5}{1.5+2} = 0.1449$$

Notice that the weights are higher for D1 than with $b = 1$.

D1 is longer than the average for our collection of 5 documents, but since the length normalization element has now been removed the weights of terms are not reduced to compensate for this.

Observations

A few observations on the example:

D3 which contains all the search terms for both search queries is scored highest by all algorithms.

The relative matching scores of the documents which partially match the query vary depending on the weighting scheme used.

We cannot see it from this simple example, but it is this variation in relative matching scores that characterizes the effectiveness of a retrieval system for a particular collection of documents. We want to select the ranking algorithm that produces consistently the “best” retrieval performance. Generating a numerical measure of retrieval effectiveness is the topic of the section of the module on Evaluation for Text Retrieval.