VECTOR SPACE

INFORMATION RETRIEVAL

The *Vector* (or *vector space*) *model* of *IR* (*VIR*) is a classical model of *IR*.

It is – traditionally – called so because both the documents and queries are conceived, for retrieval purposes, as strings of numbers as though they were (mathematical) vectors.

Retrieval is based on whether the 'query vector' and the 'document vector' are 'close enough'.

Given a finite set D of elements called *documents*:

$$D = \{D_1, ..., D_j, ..., D_m\}$$

and a finite set T of elements called *index terms*:

$$T = \{t_1, ..., t_i, ..., t_n\}$$

Any document D_j is assigned a vector $\mathbf{v}_j$ of finite real numbers, called *weights*, of length m as follows:

$$\mathbf{v}_j = (w_{ij})_{i=1,...,n} = (w_{1j}, ..., w_{ij}, ..., w_{nj})$$

where $0 \leq w_{ij} \leq 1$ (i.e. w_{ij} is normalised, e.g. division by the largest).

The weight w_{ij} is interpreted as an extent to which the term t_i 'characterises' document D_j.

The choice of terms and weights is a difficult

- theoretical (e.g. linguistic, semantical) and
- practical problem,

and several techniques can be used to cope with it.

The *VIR* model assumes that the most obvious place where appropriate content identifiers might be found are the documents themselves.

In e.g. [Luhn, 1959; Hays, 1966], it is assumed that frequencies of words in documents can give meaningful indication of their content, hence they can be taken as *identifiers*.

It is possible to elaborate different methods to compute word significance factors (or weights) related to documents.

Steps of this simple automatic method:

Step 1. Identify lexical units. Write a computer program to recognise words (word = any sequence of characters preceded and followed by ‘space’, ‘dot’, ‘comma’).

Step 2. Exclude *stopwords* from documents, i.e. those words that are unlikely to bear any significance. For example, *a*, *about*, *and*, etc.

Step 3. Apply *stemming* to the remaining words, i.e. reduce or transform them to their linguistic roots. A widely used stemming algorithm is the well-known Porter's algorithm. Practically, the index terms are the stems.

Step 4. Compute for each document D_j the number of occurrences f_{ij} of each term t_i in that document.

Step 5. Calculate the total tf_i for each term t_i as follows

$$tf_i = \Sigma_j{}^{m}{}_{=1}\ f_{ij}$$

Step 6. Rank the terms in decreasing order according to tf_i, and exclude

- the very high frequency terms, e.g. over some threshold (on the ground that they are almost always insignificant), and
- the very low frequency terms as well, e.g. below some threshold (on the ground that they are not much on the writer's mind).

Step 7. The remaining terms can be taken as document identifiers; they will be called *index terms* (or simply terms).

Step 8. Compute for the index terms t_i a *significance factor* (or *weight*) w_{ij} with respect to each document D_j. Several methods are known.

An object Q_k, called *query*, coming from a user, is also conceived as being a (much smaller) document, a vector $\mathbf{v}_k$ can be computed for it, too, in a similar way.

Retrieval is now defined:

A document D_j is retrieved in response to a query Q_k, if the document and the query are "similar enough", i.e. a similarity measure s_{jk} between

- *the document (identified by $\mathbf{v}_j$) and*
- *the query (identified by $\mathbf{v}_k$)*

is over some threshold K, i.e.

$$S_{jk} = s(\mathbf{v}_j, \mathbf{v}_k) > K$$

Steps of retrieval:

Step 9. Define query Q_k.

Step 10. Compute for the index terms t_i a *significance factor* (or *weight*) w_{ik} with respect query Q_k in the same way as the documents D_j.

Step 11. Compute the similarity values for the documents D_j.

Step 12. Give the hit list of the retrieved documents (in decreasing order) in respect to the similarity values.

Similarity measures

a) Dot product (simple matching coefficient; inner product):

$s_{jk} = (\mathbf{v}_j, \mathbf{v}_k) = \Sigma_{i=1}^{n} w_{ij} w_{ik}$

If D_j and Q_k are conceived as sets of terms, the set theoretic counterpart of the simple matching coefficient is:

$s_{jk} = |D_j \cap Q_k|$

b) Cosine measure: c_{jk}

$s_{jk} = c_{jk} = (\mathbf{v}_j, \mathbf{v}_k)/(||\mathbf{v}_j|| \cdot ||\mathbf{v}_k||)$

If D_j and Q_k are conceived as sets of terms, the set theoretic counterpart of the Cosine measure is:

$c_{jk} = |D_j \cap Q_k| \; / \; (|D_j| \cdot |Q_k|)^{1/2}$

c) Dice's coefficient: d_{jk}

$$s_{jk} = d_{jk} = 2\cdot(\mathbf{v}_j, \mathbf{v}_k) \,/\, \Sigma_{i=1}^{n} (w_{ij} + w_{ik})$$

If D_j and Q_k are conceived as sets of terms, the set theoretic counterpart of Dice's coefficient is:

$$d_{jk} = 2\cdot|D_j \cap Q_k| \,/\, (|D_j| + |Q_k|)$$

d) Jaccard's coefficient: J_{jk}

$$s_{jk} = J_{jk} = (\mathbf{v}_j, \mathbf{v}_k) \,/\, \Sigma_{i=1}^{n} (w_{ij} + w_{ik})/2^{w_{ij}w_{ik}}$$

If D_j and Q_k are conceived as sets of terms, the set theoretic counterpart of Jaccard's coefficient is:

$$J_{jk} = |D_j \cap Q_k| \,/\, |D_j \cup Q_k|$$

e) Overlap coefficient: O_{jk}

$$s_{jk} = O_{jk} = (\mathbf{v}_j, \mathbf{v}_k) \,/\, \min(\Sigma_{i=1}^{n} w_{ij},\ \Sigma_{i=1}^{n} w_{ik})$$

If D_j and Q_k are conceived as sets of terms, the set theoretic counterpart of the overlap coefficient is:

$$O_{jk} = |D_j \cap Q_k| \,/\, \min(|D_j|,|Q_k|)$$

EXAMPLE

Let the set of documents be

$$O = \{O_1, O_2, O_3\}$$

where

O_1 = *Bayes' Principle: The principle that, in estimating a parameter, one should initially assume that each possible value has equal probability (a uniform prior distribution).*

O_2 = *Bayesian Decision Theory: A mathematical theory of decision-making which presumes utility and probability functions, and according to which the act to be chosen is the Bayes act, i.e. the one with highest Subjective Expected Utility. If one had unlimited time and calculating power with which to make every decision, this procedure would be the best way to make any decision.*

O_3 = *Bayesian Epistemology: A philosophical theory which holds that the epistemic status of a proposition (i.e. how well proven or well established it is) is best measured by a probability and that the proper way to revise this probability is given by Bayesian conditionalisation or similar procedures. A Bayesian epistemologist would use probability to define, and explore the relationship between, concepts such as epistemic status, support or explanatory power.*

Step 1. Identify lexical units

Original text:

```
Bayes' Principle: The principle that,
in estimating a parameter, one should
initially assume that each possible
value has equal probability (a uniform
prior distribution).
```

Output:

Bayes	assume
principle	that
the	each
principle	possible
that	value
in	has
estimating	equal
a	probability
parameter	a
one	uniform
should	prior
initially	distribution

Step 2. Exclude stopwords from the document

Stoplist:

a	afterwards
aboard	again
about	against
above	ago
accordingly	all
across	allows
actually	almost
add	alone
added	along
after	alongside

Step 3. Apply stemming

Text before and after stemmig:

Bayes	Bayes
principle	principl
the	the
principle	principl
that	that
in	in
estimating	estim
a	a
parameter	paramet
one	on
should	should
initially	initi
assume	assum
that	that
each	each
possible	possibl
value	valu
has	ha
equal	equal
probability	probabl
a	a
uniform	uniform
prior	prior
distribution	distribut

Step 4. Compute for each document D_j the number of occurrences $\boldsymbol{f_{ij}}$ of each term t_i in that document.

E.g., t_1 := Bayes

$f_{11} = 1;$

$f_{12} = 2;$

$f_{13} = 3;$

Step 5. Calculate the total $\boldsymbol{tf_i}$ for each term t_i

number of term t_1= Bayes in all the documents:

E.g., $tf_1 = 6$

Step 6. Rank the terms in decreasing order according to tf_i, and exclude the very high and very low frequency terms

Step 7. The remaining terms can be taken as *index terms*.

Let the set T of index terms be (not stemmed here)

$T = \{t_1, t_2, t_3\} = \{$

t_1 = Bayes,

t_2 = probability,

t_3 = epistemology

$\}$.

Conceive the documents as sets of terms:

$D = \{ D_1, D_2, D_3\}$

where

$D_1 = \{(t_1$ = Bayes); (t_2 = probability)}

$D_2 = \{(t_1$ = Bayes); (t_2 = probability)}

$D_3 = \{(t_1$ = Bayes); (t_2 = probability); (t_3 = epistemology)}

Conceive the documents as sets of terms

(together with their frequencies):

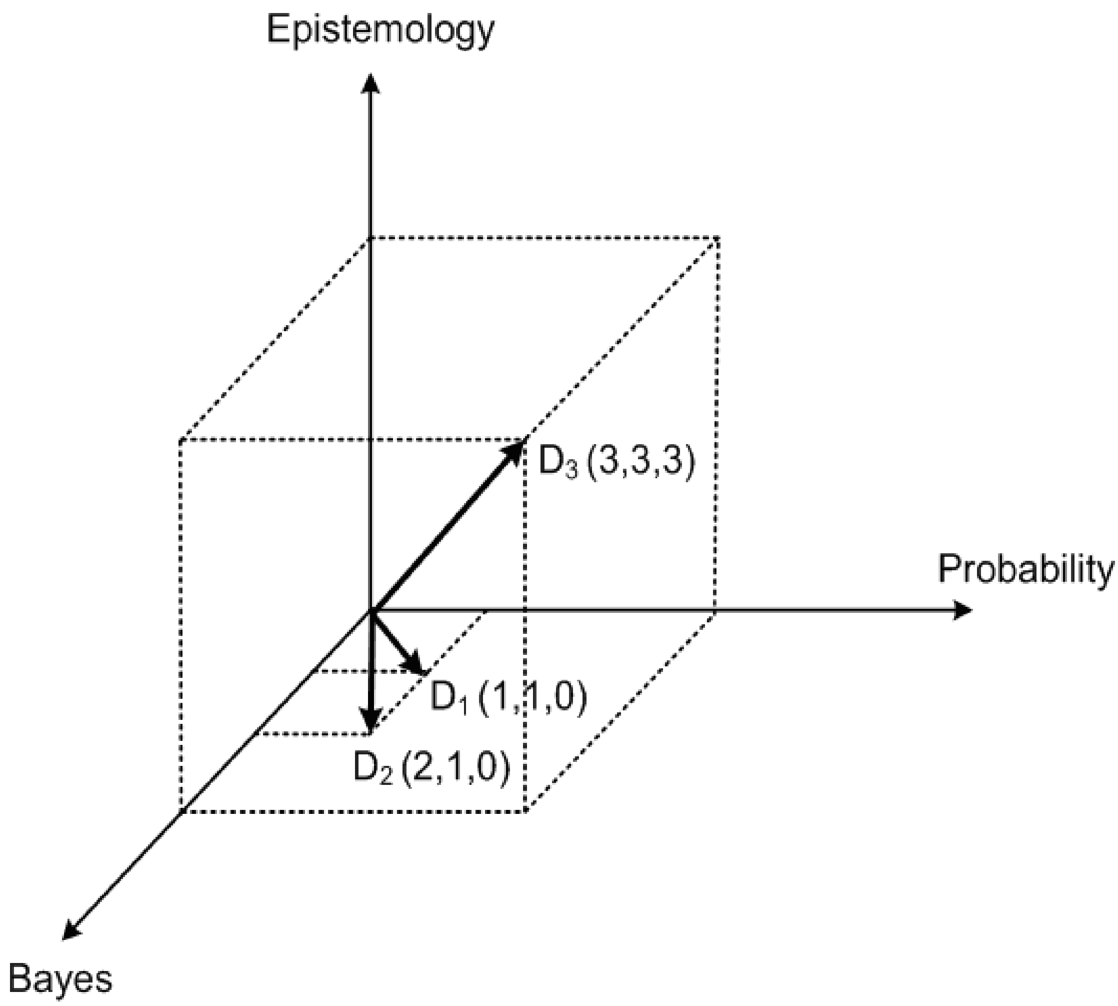
$D = \{D_1, D_2, D_3\}$

where

D_1 = { (Bayes, 1); (probability, 1); (epistemology, 0) }

D_2 = { (Bayes, 2); (probability, 1); (epistemology, 0) }

D_3 = { (Bayes, 3); (probability, 3); (epistemology, 3) }



Step 8. Compute for the index terms t_i a *significance factor* (or *weight*) w_{ij} with respect to each document D_j.

Creating Term-Document matrix: *TD:*

$TD_{3\times 3} = (w_{ij})$, where

w_{ij} denotes the weight of term t_i in document D_j

- Using *Frequency Weighting Method*:

$$TD = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 3 \\ 0 & 0 & 3 \end{pmatrix}$$

- Using *Term Frequency Normalized Weighting Method*:

$$TD = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{2\sqrt{5}}{5} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{5}}{5} & \frac{\sqrt{3}}{3} \\ 0 & 0 & \frac{\sqrt{3}}{3} \end{pmatrix}$$

Step 9. Define a query.

Let the query Q be:

Q = {What is Bayesian epistemology? }

Let the query Q be (as a set of terms):

Q = {(t_1 = Bayes); (t_3 = epistemology) }

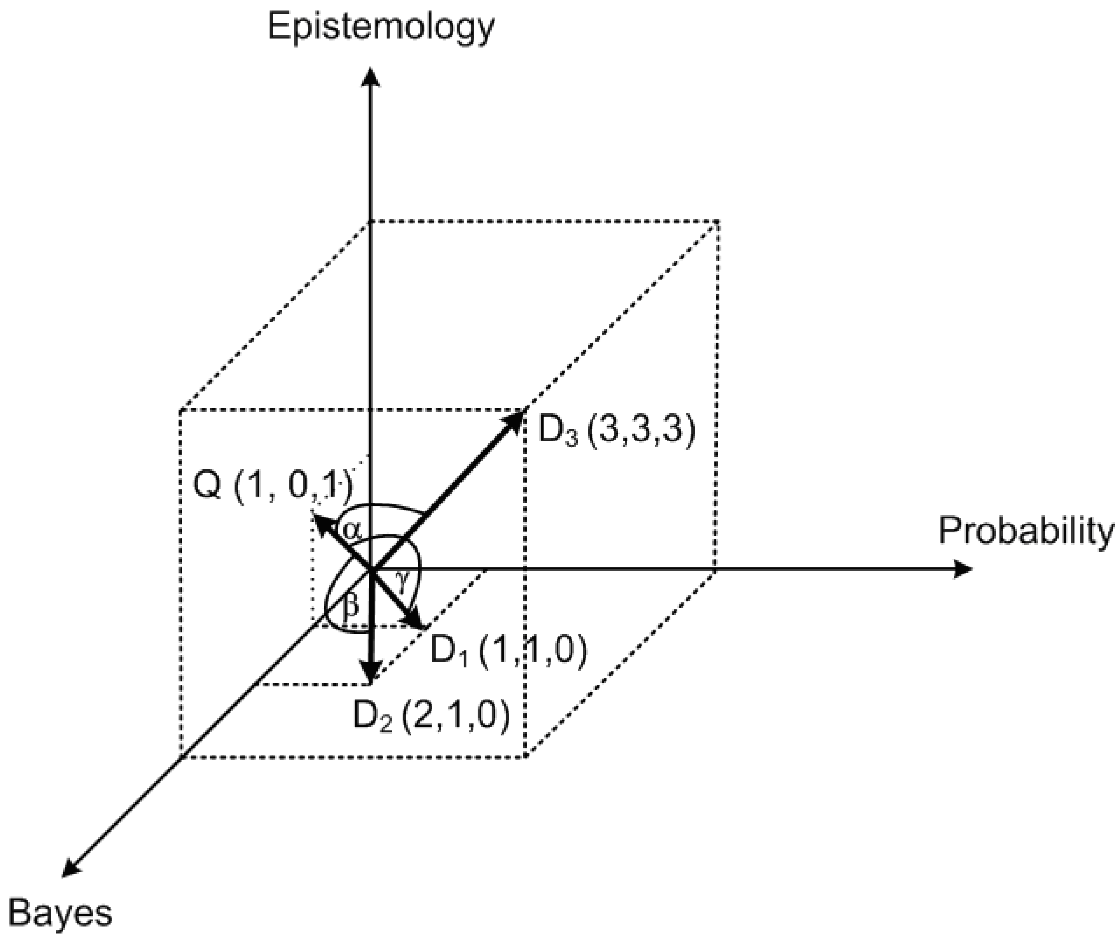
Notice that, in the *VIR*, the query is not a Boolean expression anymore (as in *BIR*).

Step 10. Compute for the index terms t_i a ***significance factor*** (or *weight*) $\boldsymbol{w_{ik}}$ with respect query Q_k.

Using
- *Frequency* or
- *Binary Weighting Method*:

$Q_k = (1, 0, 1)$

$$Q = (1, 0, 1)$$

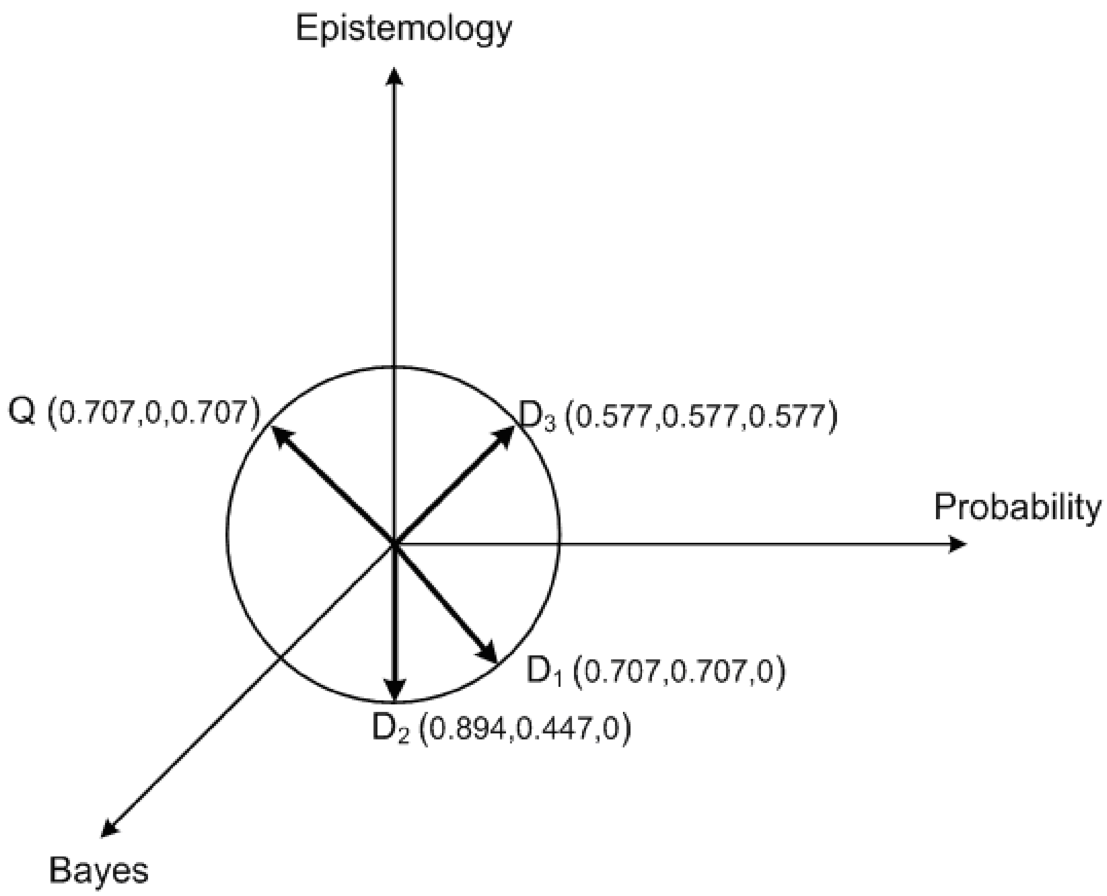


Epistemology
D3 (3,3,3)
Q (1, 0,1)
α
γ
β
Probability
D1 (1,1,0)
D2 (2,1,0)
Bayes

Step 11. -12. Compute the similarity values, and give the ranking order.

Using, *Term Frequency Normalized (tfn) Weighting Method* and

$$Q = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ 0 \\ \frac{\sqrt{2}}{2} \end{pmatrix} \qquad TD = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{2\sqrt{5}}{5} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{5}}{5} & \frac{\sqrt{3}}{3} \\ 0 & 0 & \frac{\sqrt{3}}{3} \end{pmatrix}$$



• *Dot Product:*

$Dot_1 = 0.5$; $Dot_2 = 0.632$; $Dot_3 = 0.816$

•*Cosine Measure:*

$Cosine_1 = 0.5$; $Cosine_2 = 0.632$; $Cosine_3 = 0.816$

•*Dice Measure:*

$Dice_1 = 0.354$; $Dice_2 = 0.459$; $Dice_3 = 0.519$

•*Jaccard Measure:*

$Jaccard_1 = 0.21$; $Jaccard_2 = 0.29$; $Jaccard_3 = 0.32$

$$\Downarrow$$

The ranking order: $D_3, D_2, D_1,$

Both the query and the document are represented as *vectors* of real numbers.

A similarity measure is meant to express a likeness between a query and a document.

It can be easily seen that the similarity measure typically has the following three basic properties:

- It is usually normalized, i.e. it takes on values between 0 and 1. (We shall call this property *normalization*.)
- Its value does not depend on the order in which the query and the document are considered, i.e. they are interchangeable in formulae. (We shall call this property *symmetry* or *commutativity*).
- It is maximal, i.e. equal to 1, when the query and the document are identical (exception: dot product; but notice that all the others are different normalised forms of it). (We shall call this property *reflexivity*.)

All but *Dot* are normalised and reflexive.
However, *Dot* itself can be made normalised and reflexive.

Those documents are said to be retrieved in response to a query for which the similarity measure exceeds some *threshold*.

Thus, in general, the vector *IR* can be mathematically formalized as follows:

Let D be a set of elements called *documents*. A function

$$\sigma: D \times D \to [0, 1]$$

is called a *similarity* if the following three properties a) through c) hold:

a) $0 \leq \sigma(a, b) \leq 1, \forall a, b \in D$, normalization;

b) $\sigma(a, b) = \sigma(b, a), \forall a, b \in D$, symmetry or commutativity;

c) $a = b \Rightarrow \sigma(a, b) = 1, \ a, b \in D$, reflexivity.