Vector Space Model

Mi Islita

Information Retrieval Intelligence

Your Source for Information Retrieval and Intelligence "Where Marketing Meets Science"

http://www.miislita.com

Document Linearization

- Document Linearization is the process by which a document is reduced to a stream of terms. This is usually done in two steps and as follows.

- **Markup and Format Removal** During this phase, all markup tags and special formatting are removed from the document.

- **Tokenization** During this phase, all remaining text is parsed, lowercased and all punctuation removed. Hyphenation rules must be invoked. For instance, some systems may elect to retain hyphens while others may be designed to either ignore hyphens or interpret these as spaces or as join tokens.

http://www.miislita.com/information-retrieval-tutorial/indexing.html

Weighting

- **Weighting** is the final stage in most IR indexing applications.
- Terms are weighted according to a given weighting model which may include local weighting, global weighting or both.
- If local weights are used, then term weights are normally expressed as term frequencies, *tf*.
- If global weights are used, the weight of a term is given by *IDF* values.
- The most common (and basic) weighting scheme is one in which local and global weights are used (*weight of a term* = *tf*IDF*). This is commonly referred to as *tf*IDF* weighting.

http://www.miislita.com/information-retrieval-tutorial/indexing.html

A 7 dimensional vector

Interactive query expansion modifies queries using terms from a user. Automatic query expansion expands queries automatically.

a

markup-free document text

interactive query expansion modifies queries using terms from a user automatic query expansion expands queries automatically

b

Tokenisation

interactive query expansion modifies queries terms automatic query expansion expands queries automatically

c

Stopword removal

interact queri expan
modifi queri term
automat queri
expan expand queri
automat

d

Stemming

automat	28	expand	28
expand	17	interact	17
modifi	17	queri	41
term	17		

e

Term weighting

http://www.miislita.com/information-retrieval-tutorial/indexing.html

The basics

"computer"

Loc x glob.

glob. : $\frac{N}{n}$

$$
\begin{array}{c|cccc}
 & T_1 & T_2 & \cdots & T_t \\
\hline
D_1 & d_{11} & d_{12} & \cdots & d_{1t} \\
D_2 & d_{21} & d_{22} & \cdots & d_{2t} \\
\vdots & \vdots & \vdots & & \vdots \\
\vdots & \vdots & \vdots & & \vdots \\
D_n & d_{n1} & d_{n2} & \cdots & d_{nt}
\end{array}
$$

2

3

0/1

of the

term frequency → local measure

discriminating power. → global measure

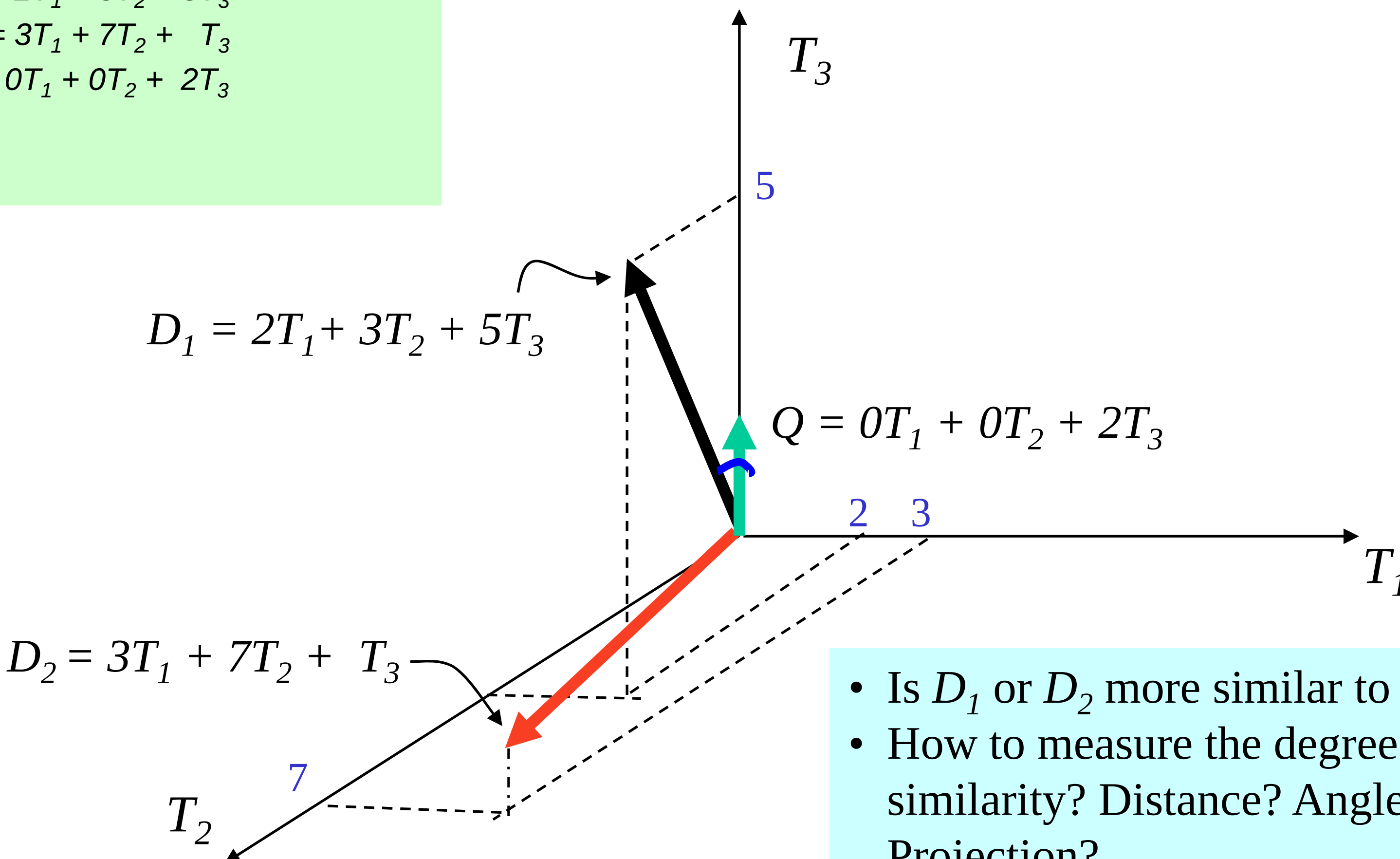
Text in a geometry: The Vector Space

Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

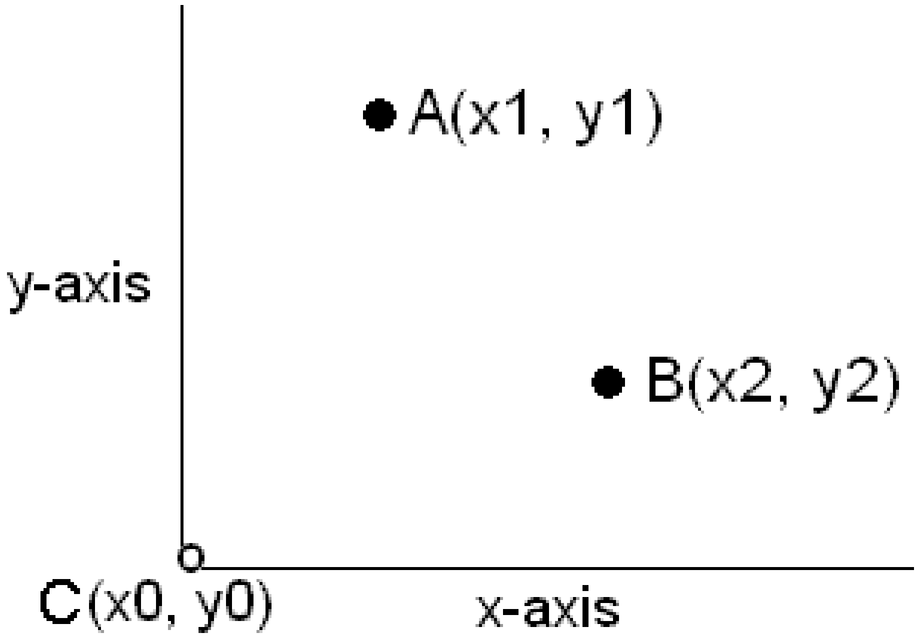
$D_2 = 3T_1 + 7T_2 + \ \ T_3$

$Q = 0T_1 + 0T_2 + \ 2T_3$



- Is D_1 or D_2 more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

The dot product



$$A \bullet B = x1*x2 + y1*y2$$

http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

Distances

idf inverse doc. frequency

global measure:

$$\boxed{\log\left(\frac{N}{n}\right)}$$

N = 1000

n = 800 the

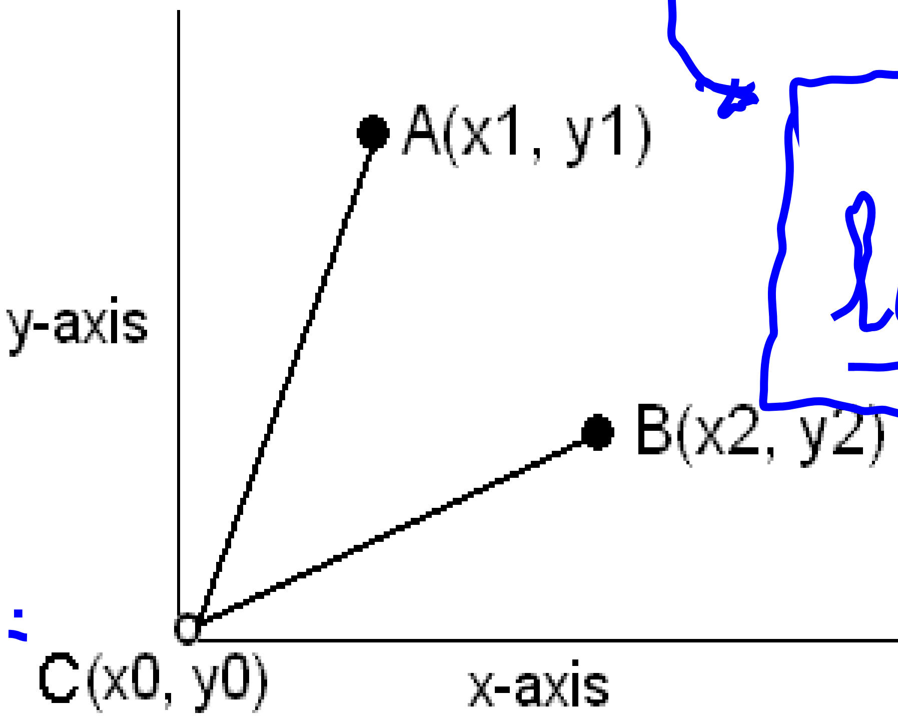
$\frac{1000}{800}$

n = 5

$\frac{1000}{5}$

200

N = total no. of docs in collection



$w_{t,d} =$

tf x idf

local

global

n = no. of docs in which the term t occurs.

$$d_{AC} = ((x_1 - x_0)^2 + (y_1 - y_0)^2)^{1/2} = (x_1^2 + y_1^2)^{1/2}$$

http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

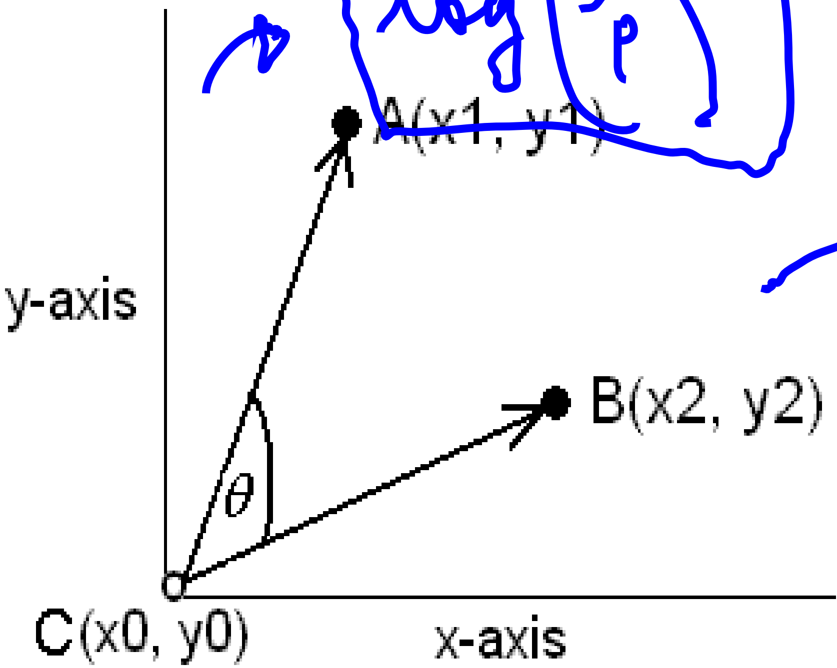
Vectors, magnitude, and cosine similarity

Information

$\log\left(\frac{1}{p}\right)$

$\log \frac{N}{n} = \log \frac{1}{[n/N]}$

p



dAC = |**A**| and dBC = |**B**|.

$$\text{Sim}(A, B) = \text{cosine } \theta = \frac{A \bullet B}{|A||B|} = \frac{x1{*}x2 + y1{*}y2}{(x1^2 + y1^2)^{1/2} \; (x2^2 + y2^2)^{1/2}}$$

http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

- This is a convenient way of ranking documents; i.e., by measuring how close their vectors are to a query vector.
- For instance, let say that point A(x1, y1) represents a query and points B(x2, y2), D(x3, y3), E(x4, y4), F(x5, y5), etc represent documents.
- We should be able to compute the cosine angle between A (the query) and each document and sort these in decreasing order of cosine angles (cosine similarites).
- This treatment can be extended to entire collection of documents.

http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

Term weighting

- By defining t_{max} as maximum term frequency in a document, N as number of documents in a collection and n as number of documents containing a query term, we can redefine term weights as

- $w = tf/t_{max}$
- $w = IDF = log(N/n)$
- $w = tf*IDF = tf*log(N/n)$
- $w = tf*IDF = tf*log((N - n)/n)$

- or even in terms of variants of *tf* and *IDF*, each one with their own customized definition and theoretical interpretation.

http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

text similarity

dot product

$$\mathrm{Sim}(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2}\sqrt{\sum_i w_{i,j}^2}}$$

magnitude of query

magnitude of i-th doc

http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

- To do this we need to construct a term space.
- The term space is defined by a list (index) of terms.
- These terms are extracted from the collection of documents to be queried.
- The coordinates of the points representing documents and queries are defined according to the weighting scheme used.
- If weights are defined as mere term counts ($w = tf$) then point coordinates are given by term frequencies;
- however, we don't have to define term weights in this manner.

http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

Global Information

Unlike the Term Count Model, Salton's Vector Space Model incorporates local and global information

$$\text{Term Weight} = \quad w_i = tf_i * \log\left(\frac{D}{df_i}\right)$$

where

tf_i = term frequency (term counts) or number of times a term i occurs in a document. This accounts for local information.

df_i = document frequency or number of documents containing term i

D = number of documents in a database.

the df_i/D ratio is the probability of selecting a document containing a queried term from a collection of documents. This can be viewed as a global probability over the entire collection. Thus, the $\log(D/df_i)$ term is the *inverse document frequency, IDF_i* and accounts for global information. If of five documents D1, D2, D3, D4, and D5, only three documents contain the term "CAR". Querying the system for this term gives an IDF value of log(5/3) = 0.2218.

http://www.miislita.com/term-vector/term-vector-3.html

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$

Query, Q: "gold silver truck"

D_1: "Shipment of gold damaged in a fire"

D_2: "Delivery of silver arrived in a silver truck"

D_3: "Shipment of gold arrived in a truck"

D = 3; IDF = $\log(D/df_i)$

	Counts, tf_i							Weights, $w_i = tf_i * IDF_i$			
Terms	Q	D_1	D_2	D_3	df_i	D/df_i	IDF_i	Q	D_1	D_2	D_3
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

http://www.miislita.com/term-vector/term-vector-3.html

Similarity Analysis

$$|D_1| = \sqrt{0.4771^2 + 0.4771^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.5173} = 0.7192$$

$$|D_2| = \sqrt{0.1761^2 + 0.4771^2 + 0.9542^2 + 0.1761^2} = \sqrt{1.2001} = 1.0955$$

$$|D_3| = \sqrt{0.1761^2 + 0.1761^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.1240} = 0.3522$$

$$\therefore \ |D_i| = \sqrt{\sum_i w_{i,j}^2}$$

$$|Q| = \sqrt{0.1761^2 + 0.4771^2 + 0.1761^2} = \sqrt{0.2896} = 0.5382$$

$$\therefore \ |Q| = \sqrt{\sum_i w_{Q,j}^2}$$

http://www.miislita.com/term-vector/term-vector-3.html

compute all dot products (zero products ignored)

$$Q \bullet D_1 = 0.1761 * 0.1761 = 0.0310$$

$$Q \bullet D_2 = 0.4771 * 0.9542 + 0.1761 * 0.1761 = 0.4862$$

$$Q \bullet D_3 = 0.1761 * 0.1761 + 0.1761 * 0.1761 = 0.0620$$

$$\therefore \; Q \bullet D_i = \sum_i w_{Q,j} w_{i,j}$$

http://www.miislita.com/term-vector/term-vector-3.html

calculate the similarity values

$$\text{Cosine } \theta_{D_1} = \frac{Q \bullet D_1}{|Q| * |D_1|} = \frac{0.0310}{0.5382 * 0.7192} = 0.0801$$

rank 3

$$\text{Cosine } \theta_{D_2} = \frac{Q \bullet D_2}{|Q| * |D_2|} = \frac{0.4862}{0.5382 * 1.0955} = 0.8246$$

rank 1

$$\text{Cosine } \theta_{D_3} = \frac{Q \bullet D_3}{|Q| * |D_3|} = \frac{0.0620}{0.5382 * 0.3522} = 0.3271$$

rank 2

$$\therefore \text{ Cosine } \theta_{D_i} = \text{Sim}(Q, D_i)$$

$$\therefore \text{ Sim}(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}}$$

Eqn. 3

http://www.miislita.com/term-vector/term-vector-3.html

Observations

- This example illustrates several facts.
- First, that very frequent terms such as "a", "in", and "of" tend to receive a low weight -a value of zero in this case.
- Thus, the model correctly predicts that very common terms, occurring in many documents in a collection are not good discriminators of relevancy. Note that this reasoning is based on global information; ie., the IDF term. Precisely, this is why this model is better than the term count model.
- Third, that instead of calculating individual vector lengths and dot products we can save computational time by applying directly the similarity function

$$\mathrm{Sim}(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}}$$

- Of course, we still need to know individual tf and IDF values.

http://www.miislita.com/term-vector/term-vector-3.html

Limitations of the Model

As a basic model, the term vector scheme discussed has several limitations.

- First, it is very calculation intensive. From the computational standpoint it is very slow, requiring a lot of processing time.
- Second, each time we add a new term into the term space we need to recalculate all vectors.
- The order in which the terms appear in the document is lost in the vector space representation

http://www.miislita.com/term-vector/term-vector-3.html

More limitations of the Model

- Long Documents: Very long documents make similarity measures difficult (vectors with small dot products and high dimensionality)
- False negative matches: documents with similar content but different vocabularies may result in a poor inner product. This is a limitation of keyword-driven IR systems.
- False positive matches: Improper wording, prefix/suffix removal or parsing can results in spurious hits (falling, fall + ing; therapist, the + rapist, the + rap + ist; Marching, March + ing; GARCIA, GAR + CIA). This is just a pre-processing limitation, not exactly a limitation of the vector model.
- Semantic content: Systems for handling semantic content may need to use special tags (containers)

http://www.miislita.com/term-vector/term-vector-3.html

We can improve the model by

- getting a set of keywords that are representative of each document.
- eliminating all stopwords and very common terms ("a", "in", "of", etc).
- stemming terms to their roots.
- limiting the vector space to nouns and few descriptive adjectives and verbs.
- using small signature files or not too huge inverted files.
- computing subvectors (passage vectors) in long documents
- not retrieving documents below a defined cosine threshold

http://www.miislita.com/term-vector/term-vector-3.html

On Polysemy and Synonymity

- A main disadvantage of this and all term vector models is that terms are assumed to be independent (i.e. no relation exists between the terms). Often this is not the case. Terms can be related by
- *Polysemy*; i.e., terms can be used to express different things in different contexts (e.g. *driving a car* and *driving results*). Thus, some irrelevant documents may have high similarities because they may share some words from the query. This affects precision.
- *Synonymity*; i.e., terms can be used to express the same thing (e.g. *car insurance* and *auto insurance*). Thus, the similarity of some relevant documents with the query can be low just because they do not share the same terms. This affects recall.
- Of these two, synonymity can produce a detrimental effect on term vector scores.

tf*idf

- More likely, no current commercial search engine implements plain *tf*idf* and for good reasons.
- One is that a raw *tf*idf* model is easy to deceive via the *tf* term.
- A keyword spammer only needs to repeat a keyword many times to increase its weight.
- This is known as *keyword spam*.
- Another reason is that term vector models assume term independence. Often, this is not the case.

Other weights - The normalized frequency

- *fi, j = tfi, j / max tfi, j*
- where
- *fi, j* = normalized frequency
  tfi, j = frequency of term i in document j
  max tfi, j = maximum frequency of term i in document j

For example, consider a document consisting of the following term counts

major, 1: league, 2: baseball, 4: playoffs, 5

Since *playoffs* occurs the most the normalized frequencies are

major, 1/5 = 0.20: league, 2/5 = 0.40: baseball, 4/5 = 0.80: playoffs, 5/5 = 1

http://www.miislita.com/term-vector/term-vector-4.html

Normalized Query Frequencies

- *fQ, i = 0.5 + 0.5*tfQ, i / max tfQ, i*

  fQ, i = normalized frequency
  tfQ, i = frequency of term i in query Q
  max tfQ, i = maximum frequency of term i in query Q

- For example, for the query *Q = major major league* the frequencies are

  major, 2
  league, 1

  since *major* occurs the most in the query, the normalized frequencies are

  major, (0.5 + 0.5*2/2) = 1
  league, (0.5 + 0.5*1/2) = 0.75

http://www.miislita.com/term-vector/term-vector-4.html

Normalized Weights

$$w_{i,j} = \frac{tf_{i,j}}{max\ tf_{i,j}} * log\left(\frac{D}{df_i}\right)$$

and the weight of term i in query Q can be written as

$$w_{Q,i} = \left(0.5 + 0.5 * \frac{tf_{Q,i}}{max\ tf_{Q,i}}\right) * log\left(\frac{D}{df_i}\right)$$

http://www.miislita.com/term-vector/term-vector-4.html

The Glasgow Model

$$w_{ij} = \frac{\log(freq_{ij} + 1)}{\log(length_j)} \bullet \log\left(\frac{N}{n_i}\right)$$

w_{ij} = tf•idf weight of term i in document j

$freq_{ij}$ = frequency of term i in document j

$length_j$ = number of unique terms in document j

N = number of documents in collection

n_i = number of documents term i occurs in

http://www.miislita.com/term-vector/term-vector-4.html