정보검색 기본이론

구명완교수

서강대학교 컴퓨터공학과

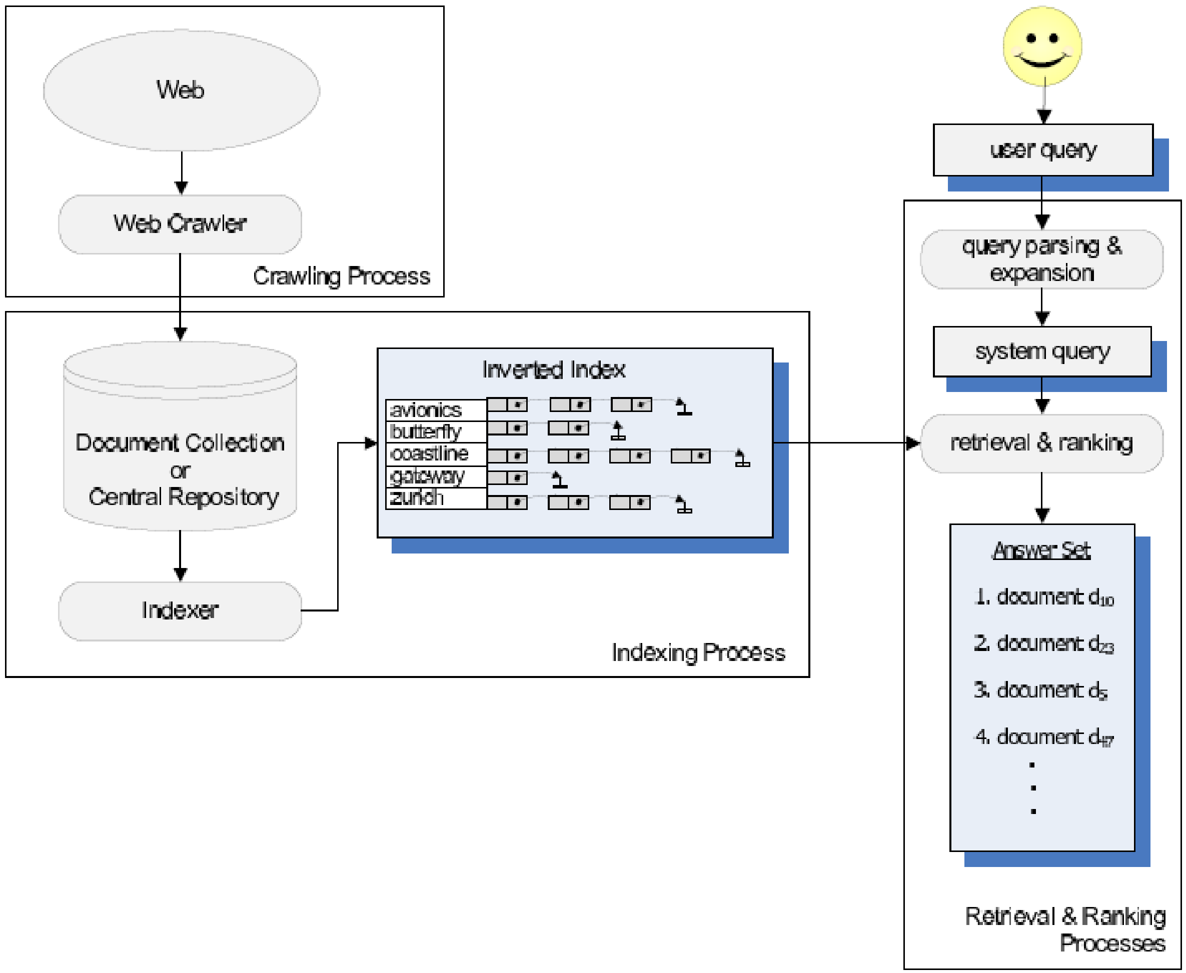
Email: mwkoo@sogang.ac.kr

Contents

- Introduction to IR Models
- Basic Concepts
- Term Weighting
- The Vector Model
- Probabilistic Model

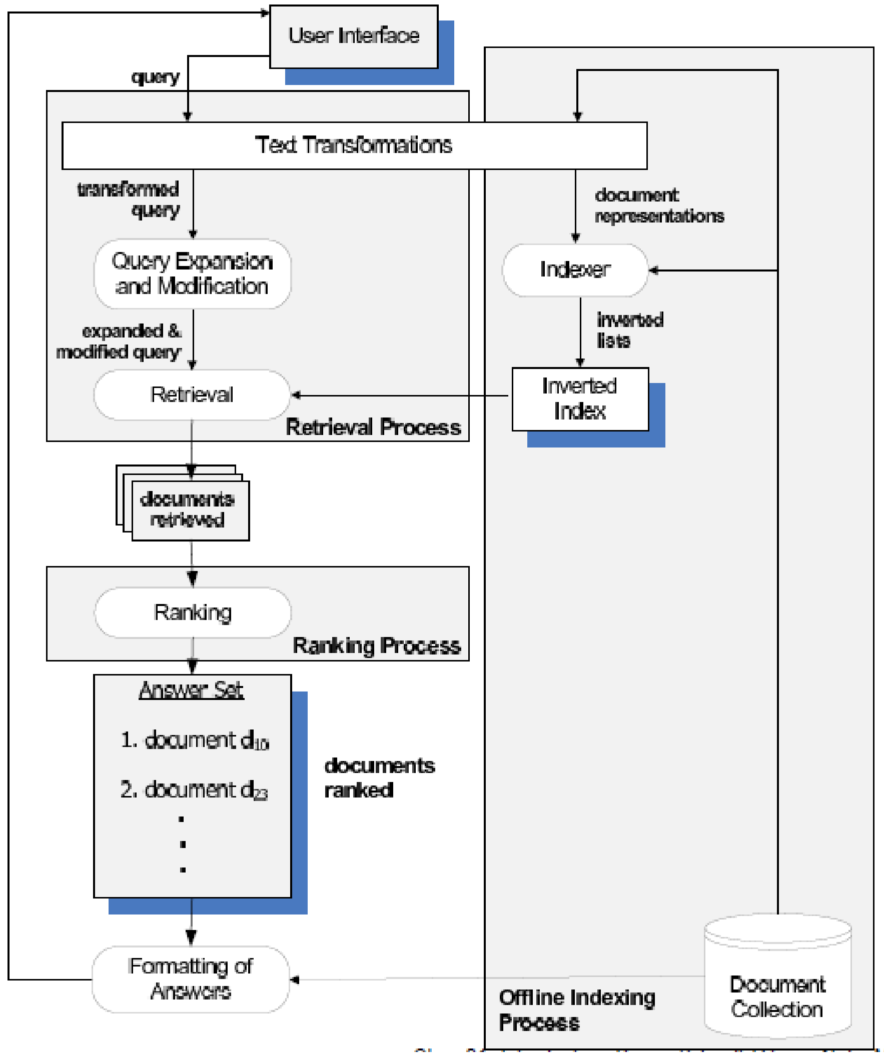
Architecture of the IR System

- High level software architecture of an IR system

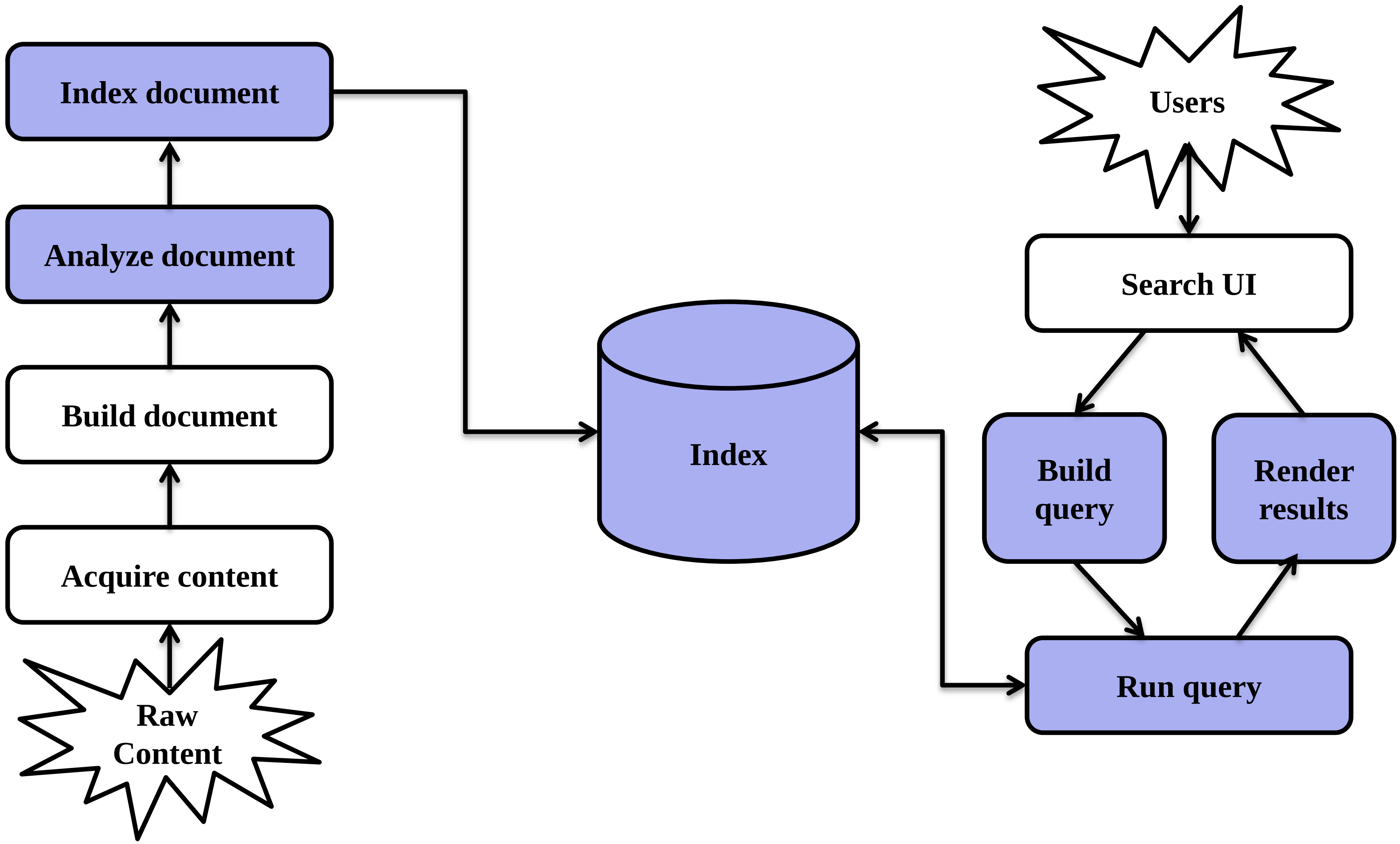


Retrieval and Ranking Processes

- The processes of *indexing*, *retrieval*, and *ranking*



Lucene in a search system



IR Models

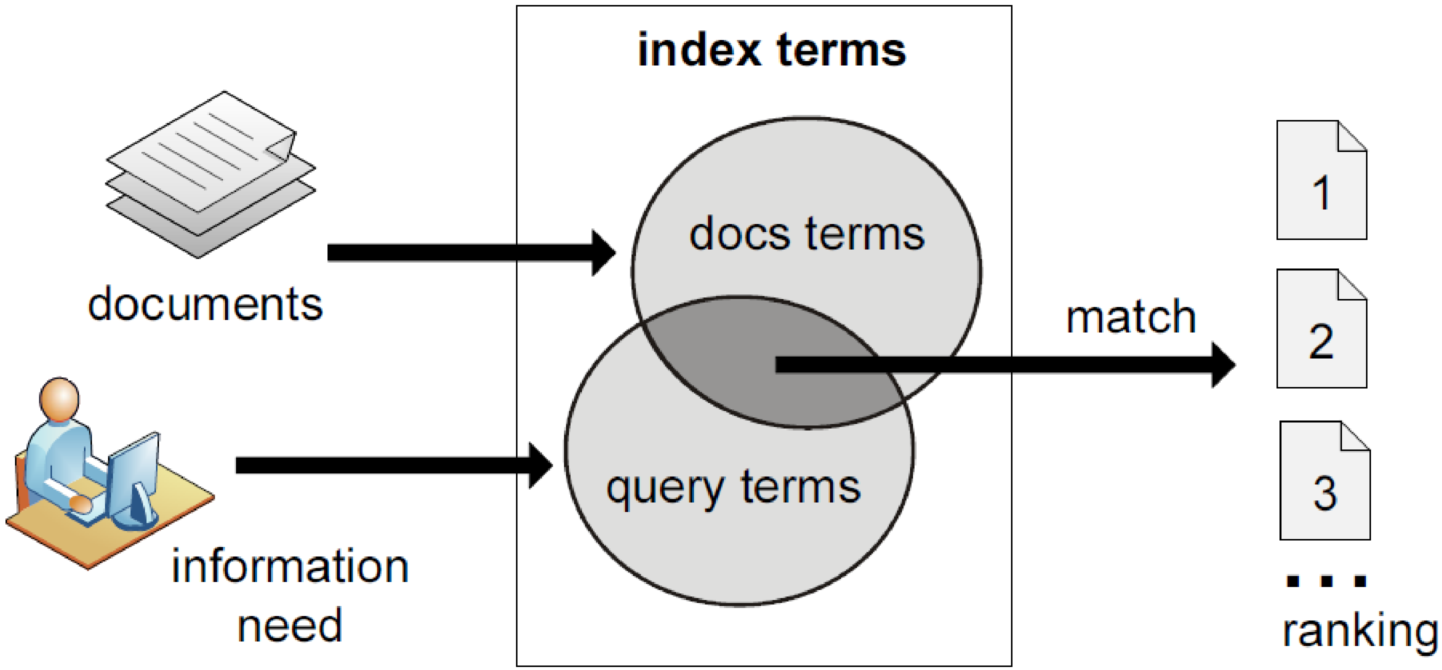
- **Modeling** in IR is a complex process aimed at producing a ranking function
  - **Ranking function**: a function that assigns scores to documents with regard to a given query
- This process consists of two main tasks:
  - The conception of a logical framework for representing documents and queries
  - The definition of a ranking function that allows quantifying the similarities among documents and queries

Modeling and Ranking

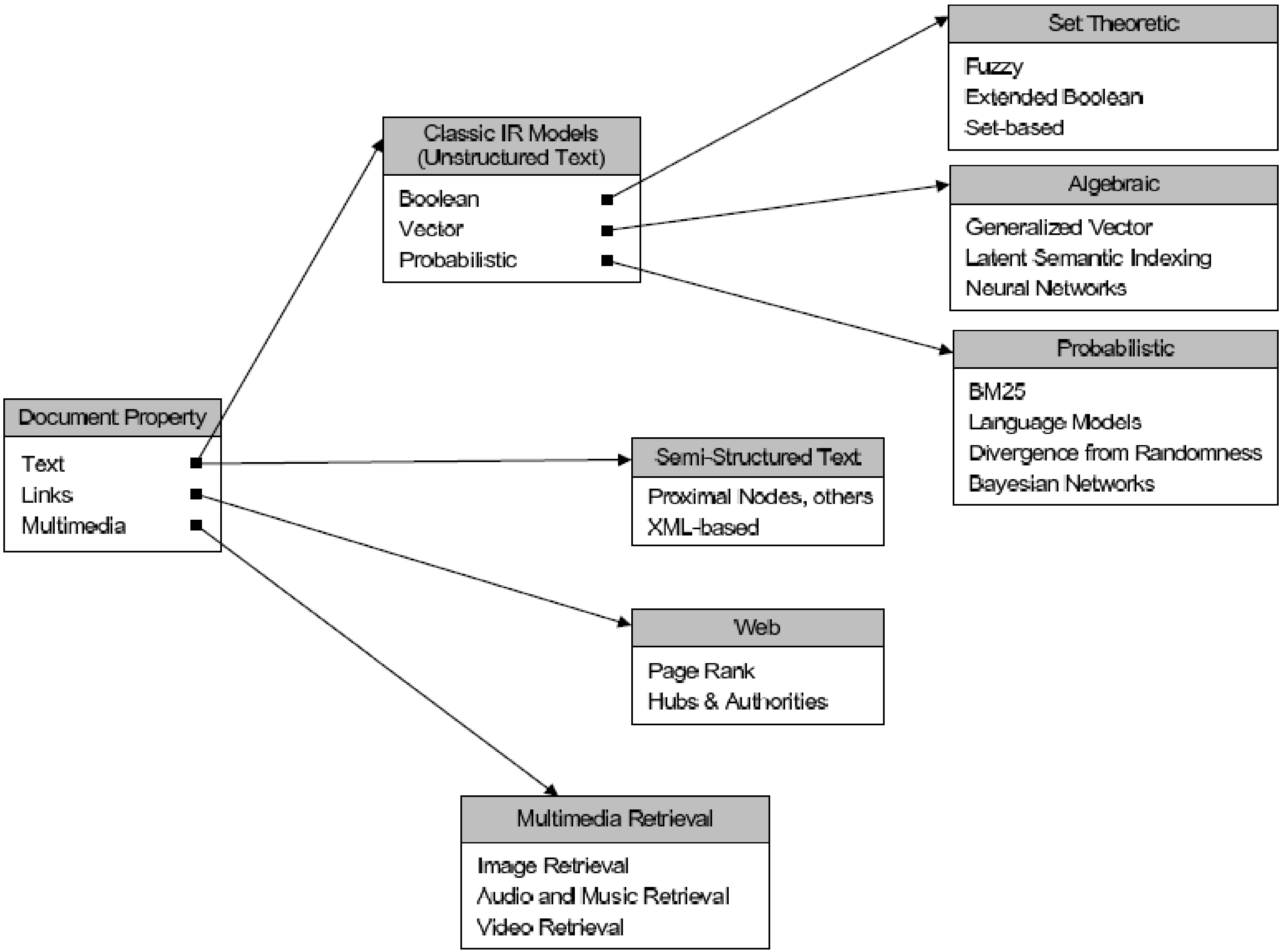
- IR systems usually adopt **index terms** to index and retrieve documents
- Index term:
  - In a restricted sense: it is a keyword that has some meaning on its own; usually plays the role of a noun
  - In a more general form: it is any word that appears in a document
- Retrieval based on index terms can be implemented efficiently
- Also, index terms are simple to refer to in a query
- Simplicity is important because it reduces the effort of query formulation

Introduction

- Information retrieval process



A Taxonomy of IR Models



Basic Concepts

- Each document is represented by a set of representative keywords or index terms
- An index term is a word or group of consecutive words in a document
- A pre-selected set of index terms can be used to summarize the document contents
- However, it might be interesting to assume that all words are index terms (full text representation)

Basic Concepts

- Let,
  - t be the number of index terms in the document collection
  - k_i be a generic index term
- Then,
  - The **vocabulary** V = $\{k_1, \ldots, k_t\}$ is the set of all distinct index terms in the collection

$$V = \boxed{k_1 \quad k_2 \quad k_3 \quad \cdots \quad k_t}$$

vocabulary of *t* index terms

Basic Concepts

- Documents and queries can be represented by **patterns of term co-occurrences**

$V=$	k_1	k_2	k_3	…	k_t	
	1	0	0	…	0	pattern that represents documents (and queries) with the term k_1 and no other
			⋮			
	1	1	1	…	1	pattern that represents documents (and queries) with all index terms

- Each of these patterns of term co-occurrence is called a **term conjunctive component**
- For each document d_j (or query q) we associate a unique term conjunctive component c(d_j) (or c(q))

The Term-Document Matrix

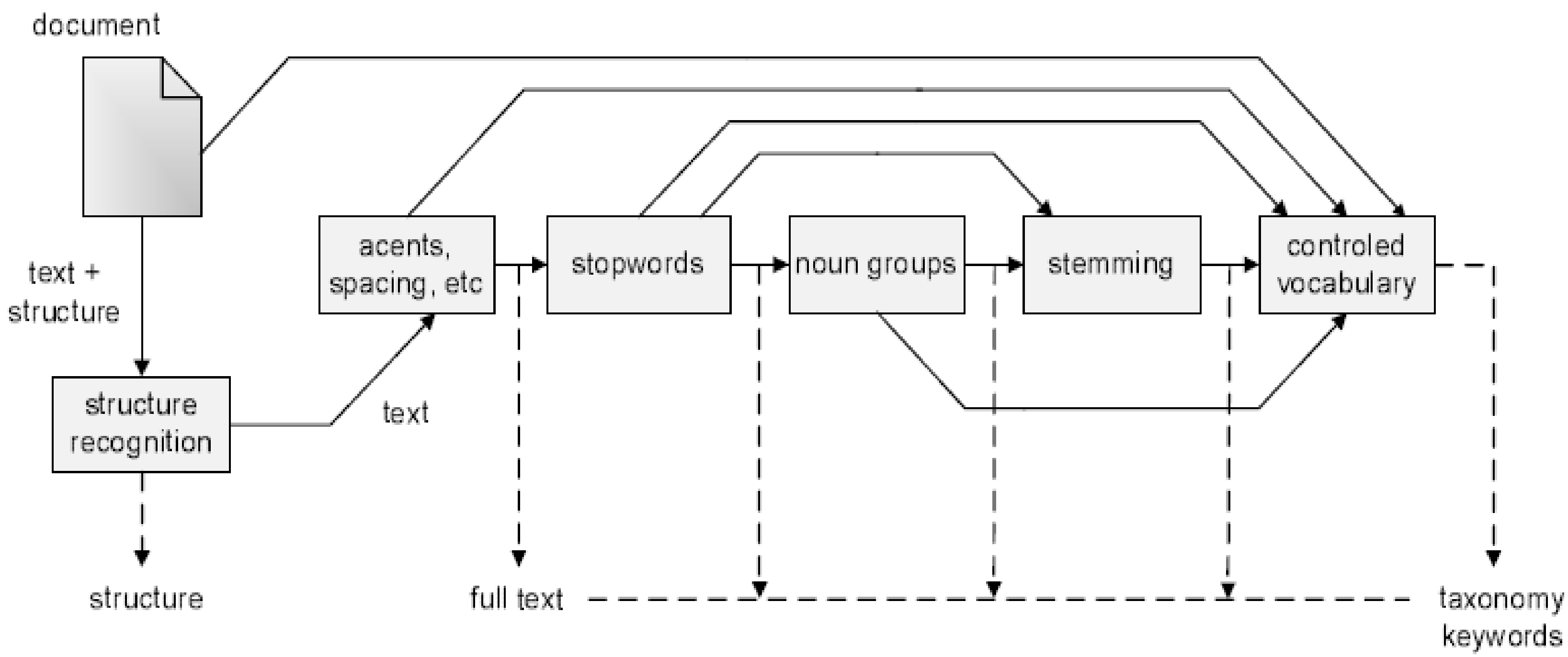
- The occurrence of a term k_i in a document d_j establishes a relation between k_i and d_j
- A **term-document relation** between k_i and d_j can be quantified by the frequency of the term in the document
- In matrix form, this can written as

$$\begin{array}{cc} & \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{array}{c} k_1 \\ k_2 \\ k_3 \end{array} & \begin{bmatrix} f_{1,1} & f_{1,2} \\ f_{2,1} & f_{2,2} \\ f_{3,1} & f_{3,2} \end{bmatrix} \end{array}$$

- where each $f_{i,j}$ element stands for the frequency of term k_i in document d_j

Basic Concepts

- Logical view of a document: from full text to a set of index terms

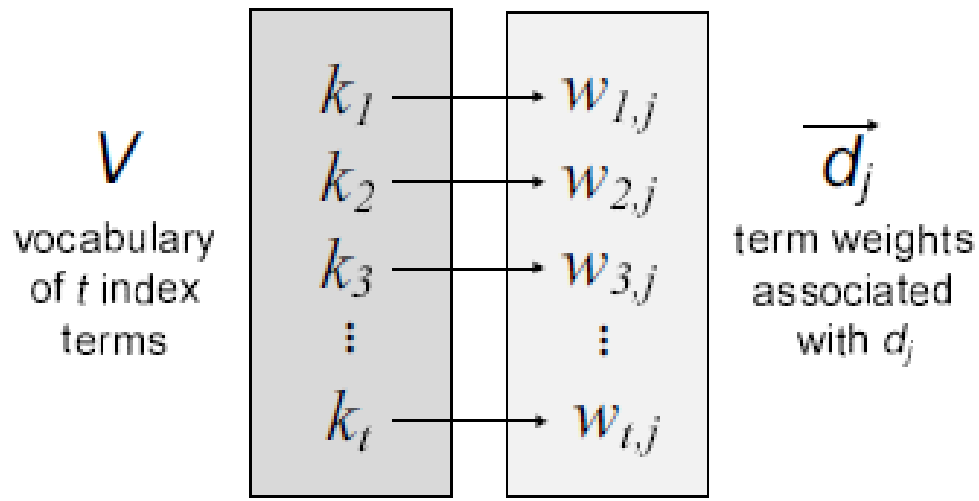


Term Weighting

- The terms of a document are not equally useful for describing the document contents
- There are properties of an index term which are useful for evaluating the importance of the term in a document
  - For instance, a word which appears in all documents of a collection is completely useless for retrieval tasks
- To characterize term importance, we associate a weight $w_{i,j} > 0$ with each term k_i that occurs in the document d_j
  - If k_i that does not appear in the document d_j , then $w_{i,j} = 0$.
- The weight $w_{i,j}$ quantifies the importance of the index term k_i for describing the contents of document d_j

Term Weighting

- Let,
  - k_i be an index term and d_j be a document
  - $V = \{ k_1, k_2, \ldots\ k_t\}$ be the set of all index terms
  - $w_{i,j} > 0$ be the weight associated with $(k_i\ , d_j\ )$
- Then we define $\overrightarrow{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$ as a weighted vector that contains the weight $w_{i,j}$ of each term $w_{i,j}\ \in V$ in the document d_j



Term Weighting

- The weights $w_{i,j}$ can be computed using the **frequencies of occurrence** of the terms within documents
- Let $f_{i,j}$ be the frequency of occurrence of index term k_i in the document d_j
- The **total frequency of occurrence** F_i of term k_i in the collection is defined as

$$F_i = \sum_{j=1}^{N} f_{i,j}$$

where N is the number of documents in the collection

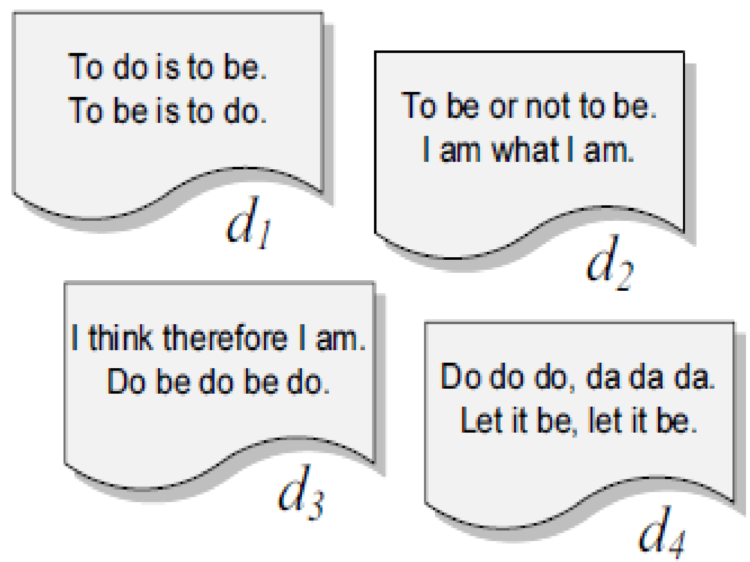
Term Weighting

- The **document frequency** $\boldsymbol{n_i}$ of a term $\boldsymbol{k_i}$ is the number of documents in which it occurs
  - Notice that $\boldsymbol{n_i} \leq \boldsymbol{F_i}$
- For instance, in the document collection below, the values $\boldsymbol{f_{i,j}}$, $\boldsymbol{F_i}$ and $\boldsymbol{n_i}$ associated with the term *do* are

$$f(do, d_1) = 2$$
$$f(do, d_2) = 0$$
$$f(do, d_3) = 3$$
$$f(do, d_4) = 3$$

$$F(do) = 8$$

$$n(do) = 3$$



TF-IDF Weights

- TF-IDF term weighting scheme:
  - Term frequency (TF)
  - Inverse document frequency (IDF)
  - Foundations of the most popular term weighting scheme in IR

Term Frequency (TF) Weights

- TF Weights(*tf*) is

$$tf_{i,j} = f_{i,j}$$

- A variant of *tf* weight used in the literature is

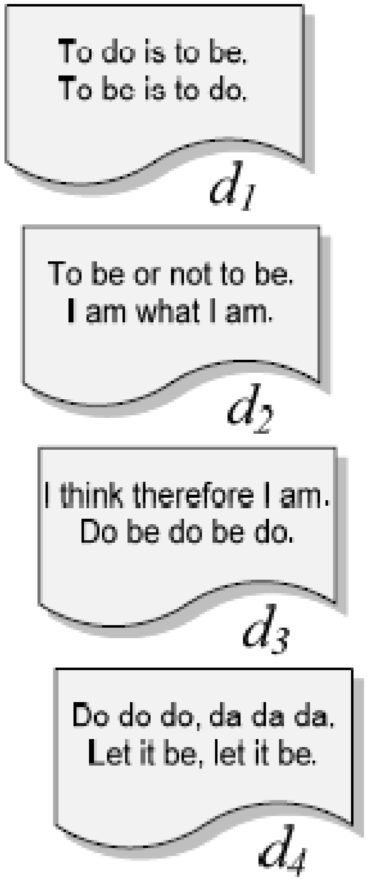
$$tf_{i,j} = \begin{cases} 1 + \log f_{i,j} & if \ f_{i,j} > 0 \\ 0 & Otherwise \end{cases}$$

where the log is taken in base 2

- The log expression is a the preferred form because it makes them directly comparable to *idf* weights, as we later discuss

Term Frequency (TF) Weights

- Log *tf* weights $tf_{i,j}$ for the example collection



Vocabulary		$tf_{i,1}$	$tf_{i,2}$	$tf_{i,3}$	$tf_{i,4}$
1	to	3	2	-	-
2	do	2	-	2.585	2.585
3	is	2	-	-	-
4	be	2	2	2	2
5	or	-	1	-	-
6	not	-	1	-	-
7	I	-	2	2	-
8	am	-	2	1	-
9	what	-	1	-	-
10	think	-	-	1	-
11	therefore	-	-	1	-
12	da	-	-	-	2.585
13	let	-	-	-	2
14	it	-	-	-	2

Inverse Document Frequency

- **Specificity** is a property of the term semantics
  - A term is more or less specific depending on its meaning
  - To exemplify, the term *beverage* is less specific than the terms *tea* and *beer*
  - We could expect that the term *beverage* occurs in more documents than the terms *tea* and *beer*
- Term specificity should be interpreted as a statistical rather than semantic property of the term
- **Statistical term specificity**. The inverse of the number of documents in which the term occurs

Inverse Document Frequency

- **Inverse document frequency** of term k_i is

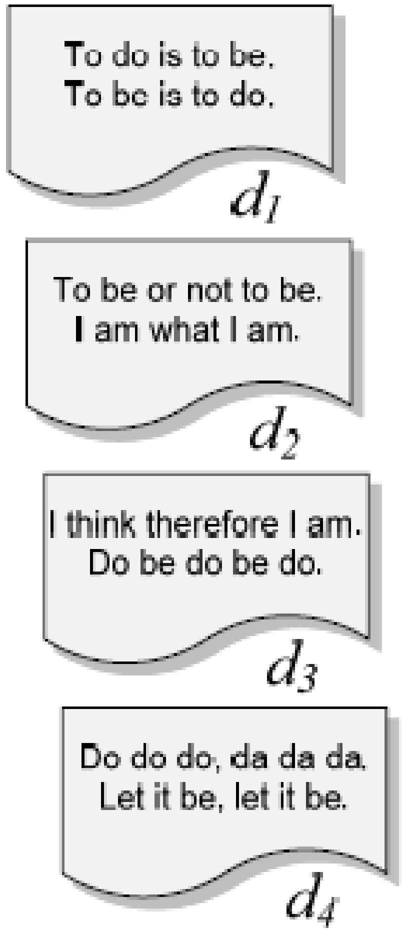
$$idf_i = \log \frac{N}{n_i}$$

where N is the number of docs in the collection and n_i is the number of docs with term k_i

- *Idf* provides a foundation for modern term weighting schemes and is used for ranking in almost all IR systems

Inverse Document Frequency

- Idf values for example collection



	term	n_i	$idf_i = \log(N/n_i)$
1	to	2	1
2	do	3	0.415
3	is	1	2
4	be	4	0
5	or	1	2
6	not	1	2
7	I	2	1
8	am	2	1
9	what	1	2
10	think	1	2
11	therefore	1	2
12	da	1	2
13	let	1	2
14	it	1	2

TF-IDF weighting scheme

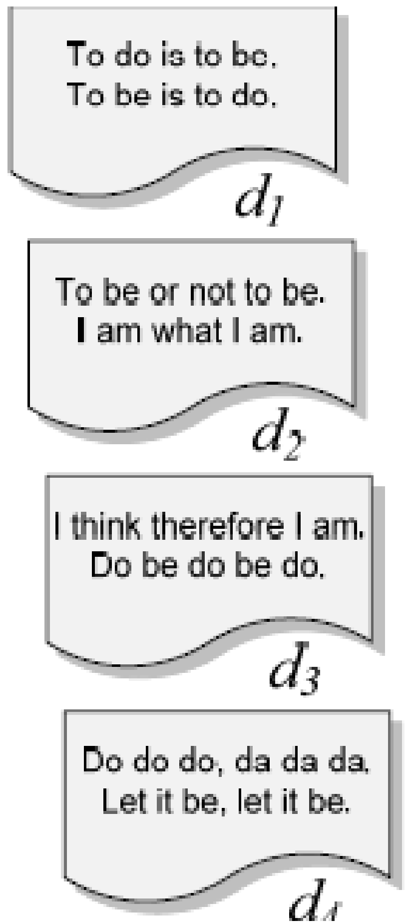
- The best known term weighting schemes use weights that combine idf factors with term frequencies
- Let $w_{i,j}$ be the term weight associated with the term k_i and the document d_j
- Then, we define

$$w_{i,j} = \begin{cases} (1 + \log f_{i,j}) \times \log \frac{N}{n_i} & \texttt{if} \quad f_{i,j} > 0 \\ 0 & \texttt{otherwise} \end{cases}$$

which is referred to as a **tf-idf weighting scheme**

TF-IDF weighting scheme

- Tf-idf weights of all terms present in our example document collection



		d_1	d_2	d_3	d_4
1	to	3	2	-	-
2	do	0.830	-	1.073	1.073
3	is	4	-	-	-
4	be	-	-	-	-
5	or	-	2	-	-
6	not	-	2	-	-
7	I	-	2	2	-
8	am	-	2	1	-
9	what	-	2	-	-
10	think	-	-	2	-
11	therefore	-	-	2	-
12	da	-	-	-	5.170
13	let	-	-	-	4
14	it	-	-	-	4

Variants of TF-IDF

- Several variations of the above expression for tf-idf weights are described in the literature
- For tf weights, five distinct variants are illustrated below

	tf weight
binary	{0,1}
raw frequency	$f_{i,j}$
log normalization	$1 + \log f_{i,j}$
double normalization 0.5	$0.5 + 0.5 \frac{f_{i,j}}{max_i f_{i,j}}$
double normalization K	$K + (1 - K) \frac{f_{i,j}}{max_i f_{i,j}}$

Variants of TF-IDF

- Five distinct variants of idf weight

	idf weight
unary	1
inverse frequency	$\log \frac{N}{n_i}$
inv frequency smooth	$\log(1 + \frac{N}{n_i})$
inv frequeny max	$\log(1 + \frac{max_i n_i}{n_i})$
probabilistic inv frequency	$\log \frac{N - n_i}{n_i}$

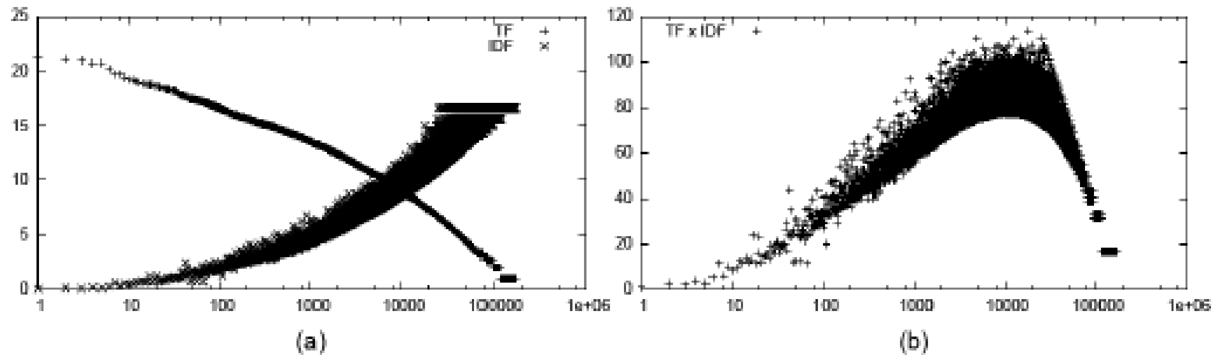
Variants of TF-IDF

- Recommended tf-idf weighting schemes

weighting scheme	document term weight	query term weight
1	$f_{i,j} * \log \frac{N}{n_i}$	$(0.5 + 0.5 \frac{f_{i,q}}{max_i \ f_{i,q}}) * \log \frac{N}{n_i}$
2	$1 + \log f_{i,j}$	$\log(1 + \frac{N}{n_i})$
3	$(1 + \log f_{i,j}) * \log \frac{N}{n_i}$	$(1 + \log f_{i,q}) * \log \frac{N}{n_i}$

TF-IDF Properties

- the tf, idf, and tf-idf weights for the *Wall Street Journal*
-



- We observe that tf and idf weights present power-law behaviors that balance each other
- The terms of intermediate idf values display maximum tf-idf weights and are most interesting for ranking

Document Length Normalization

- Document sizes might vary widely
- This is a problem because longer documents are more likely to be retrieved by a given query
- To compensate for this undesired effect, we can divide the rank of each document by its length
- This procedure consistently leads to better ranking, and it is called **document length normalization**

Document Length Normalization

- Methods of document length normalization depend on the representation adopted for the documents:
  - **Size in bytes**: consider that each document is represented simply as a stream of bytes
  - **Number of words**: each document is represented as a single string, and the document length is the number of words in it
  - **Vector norms**: documents are represented as vectors of weighted terms

Document Length Normalization

- The document representation $\overrightarrow{d_j}$ is a vector composed of all its term vector components

$$\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$$

- The document length is given by the norm of this vector, which is computed as follows

$$|\vec{d_j}| = \sqrt{\sum_{i}^{t} w_{i,j}^2}$$

The Vector Model

- Boolean matching and binary weights is too limiting
- The vector model proposes a framework in which partial matching is possible
- This is accomplished by assigning non-binary weights to index terms in queries and in documents
- Term weights are used to compute a **degree of similarity** between a query and each document
- The documents are **ranked** in decreasing order of their degree of similarity

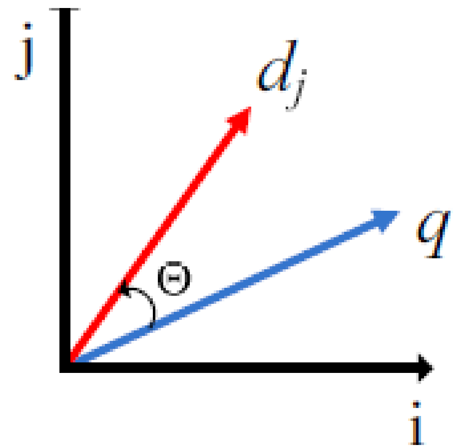
The Vector Model

- For the vector model:
  - The weight $w_{i,j}$ associated with a pair (k_i, d_j) is positive and non-binary
  - The index terms are assumed to be all mutually independent
  - They are represented as unit vectors of a *t*-dimensional space (*t* is the total number of index terms)
  - The representations of document d_j and query q are t-dimensional vectors given by

$$\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{tj})$$
$$\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{tq})$$

The Vector Model

- Similarity between a document d_j and a query q



$$cos(\theta) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|}$$

$$\boxed{sim(d_j, q) = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}}$$

Since $w_{ij} > 0$ and $w_{iq} > 0$, we have $0 \leqslant sim(d_j, q) \leqslant 1$

The Vector Model

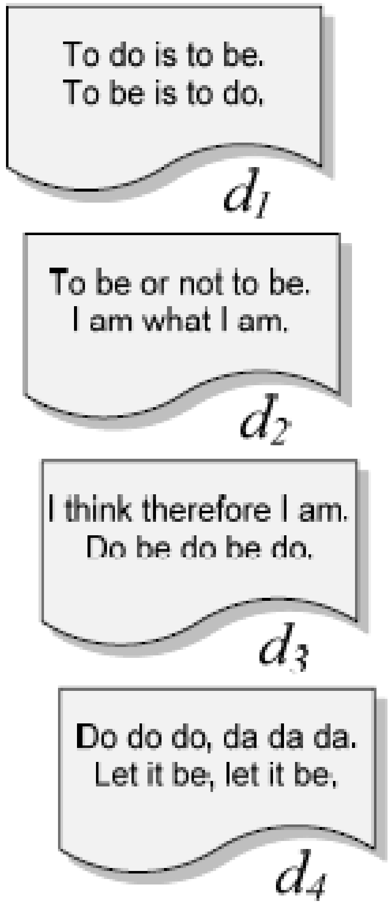
- Weights in the Vector model are basically tf-idf weights

$$
\begin{aligned}
w_{i,q} &= (1 + \log f_{i,q}) \times \log \frac{N}{n_i} \\
w_{i,j} &= (1 + \log f_{i,j}) \times \log \frac{N}{n_i}
\end{aligned}
$$

- These equations should only be applied for values of term frequency greater than zero
- If the term frequency is zero, the respective weight is also zero

The Vector Model

- Document ranks computed by the Vector model for the query "to do"



doc	rank computation	rank
d_1	$\frac{1*3+0.415*0.830}{5.068}$	0.660
d_2	$\frac{1*2+0.415*0}{4.899}$	0.408
d_3	$\frac{1*0+0.415*1.073}{3.762}$	0.118
d_4	$\frac{1*0+0.415*1.073}{7.738}$	0.058

The Vector Model

- Advantages:
  - term-weighting improves quality of the answer set
  - partial matching allows retrieval of docs that approximate the query conditions
  - cosine ranking formula sorts documents according to a degree of similarity to the query
  - document length normalization is naturally built-in into the ranking
- Disadvantages:
  - It assumes independence of index terms

Probabilistic Model

- The probabilistic model captures the IR problem using a probabilistic framework
- Given a user query, there is an **ideal answer set** for this query
- Given a description of this ideal answer set, we could retrieve the relevant documents
- Querying is seen as a specification of the **properties** of this ideal answer set
- But, what are these properties?

Ranking Formula

$$sim(d_j, q) \sim \sum_{k_i[q,d_j]} \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

- This equation provides an idf-like ranking computation
- Neither TF weights Nor document length normalization
  - BM25 model

Comparison of Classic Models

- Boolean model does not provide for partial matches and is considered to be the weakest classic model
- There is some controversy as to whether the probabilistic model outperforms the vector model
- Croft suggested that the probabilistic model provides a better retrieval performance
- However, Salton *et al* showed that the vector model outperforms it with general collections
  - This also seems to be the dominant thought among researchers and practitioners of IR.

Alternative Probabilistic Models

- **BM25 Ranking Formula (Lucene 5.0)**

$$sim_{BM25}(d_j, q) \sim \sum_{k_i[q,d_j]} \mathcal{B}_{i,j} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

$$\mathcal{B}_{i,j} = \frac{(K_1 + 1) f_{i,j}}{K_1 \left[(1 - b) + b \frac{len(d_j)}{avg\_doclen}\right] + f_{i,j}}$$

- Adding term frequency factor & document length normalization by experiment (K_1=1, 0 ≤b ≤ 1)
- BM25 outperforms classic vector model for general collections
- Used as a baseline for evaluating new ranking functions, in substitution to the classic vector model

Link-based Ranking(PageRank)

- The basic idea is that good pages point to good pages
- Let p, r be two variables for pages and L a set of links

PageRank Algorithm

1. $p :=$ initial page the user is at;
2. while (stop-criterion is not met) {
3. $\quad L := links\_inside\_page(p)$;
4. $\quad r := random(L)$;
5. $\quad$ move to page pointed by r;
6. $\quad p := r$;
7. }

Link-based Ranking(PageRank)

- Notice that PageRank simulates a user navigating randomly on the Web
- At infinity, the probability of finding the user at any given page becomes stationary
- Process can be modeled by a Markov chain
  - stationary probability of being at each page can be computed
- This probability is a property of the graph
  - referred to as **PageRank** in the context of the Web
- PageRank is the best known link-based weighting scheme
- It is also part of the ranking strategy adopted by Google

Link-based Ranking(PageRank)

- Let
  - Let $L(p)$ be the number of outgoing links of page p
  - Let $p_1 \ldots p_n$ be the pages that point to page a
  - User jumps to a random page with probability q
  - User follows one of the links in current page with probability $1 - q$
  - **PageRank** of page a is given by the probability $PR(a)$ of finding our user in that page

$$PR(a) = \frac{q}{T} + (1 - q) \sum_{i=1}^{n} \frac{PR(p_i)}{L(p_i)}$$

where

- T: total number of pages on the Web graph
- q: parameter set by the system (typical value is 0.15)

Simple Ranking Functions

- More elaborate ranking scheme
  - use a linear combination of different ranking signals
  - for instance, combine BM25 (text-based ranking) with PageRank (link-based ranking)
- Rank score $R(p, Q)$ of page p with regard to query Q can be computed as

$$R(p, Q) = \alpha\ BM25(p, Q)\ +\ (1 - \alpha) PR(p)$$

  - $\alpha = 1$: text-based ranking, early search engines
  - $\alpha = 0$: link-based ranking, independent of the query