Information Retrieval

D.Thenmozhi

SSN College of Engineering

March 29, 2017

Overview

Introduction

Definition of IR

Information retrieval deals with the organisation, storage, retrieval and evaluation of information relevant to a user's query.

1. The type of **information-items** include - documents, web pages, online catalogs and multimedia objects.
2. A user in need of information formulates a request in the form of a query written in natural language
3. Example of IR System : Web Search Engine

IR Problem 1 : **Nature of User Query**

Consider the following query :

"Find all the documents that address the role of Indian Government, in financing the operation of the National Railroad Corporation"

- This full description of the user need is not necessarily to be submitted to IR system.
- First, you need to translate the natural language information into a query.
- This must contain a set of keywords and index terms.

IR Problem 2 : **Notion of relevance**

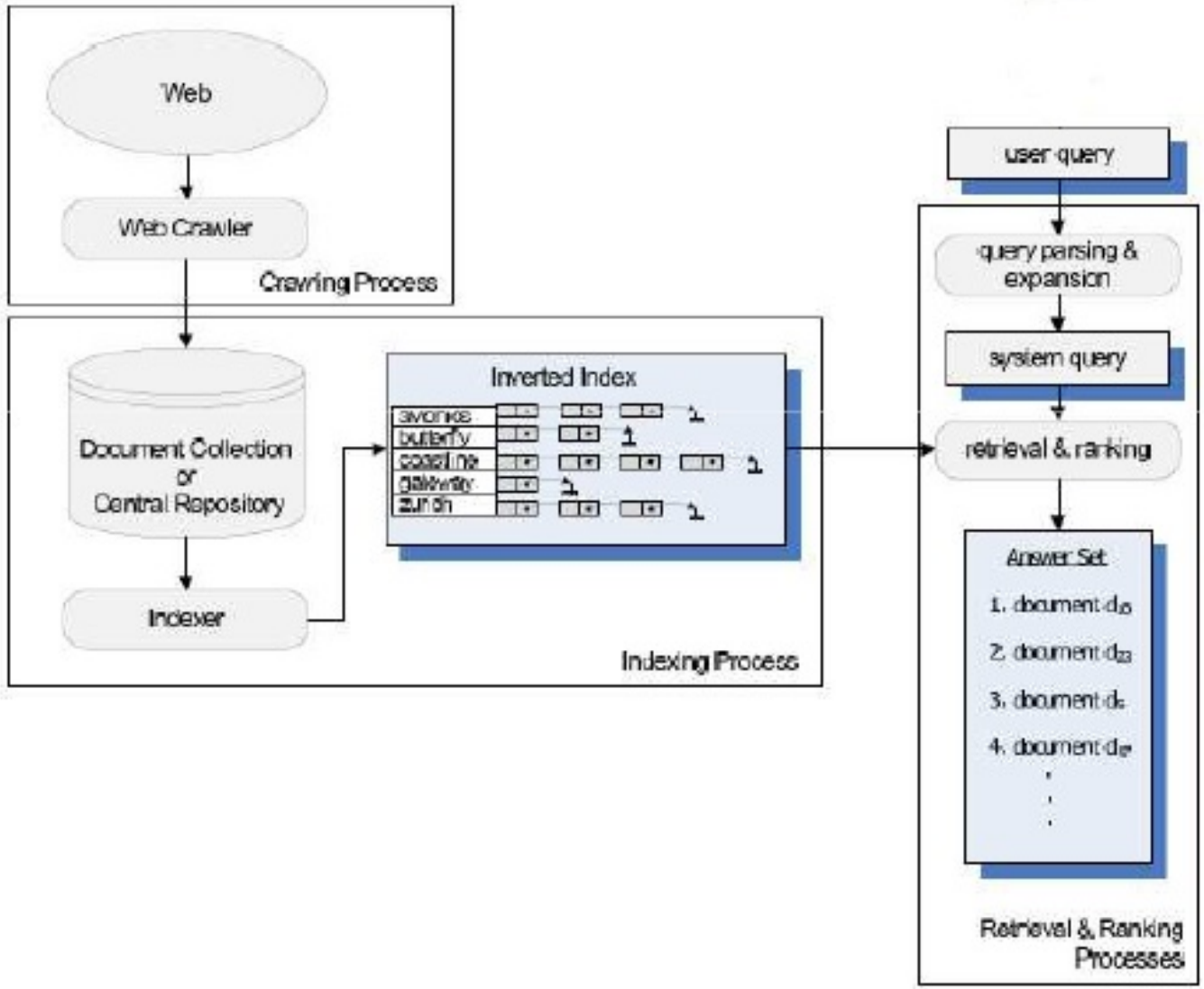
The IR System must rank the "information-items" according to the degree of relevance to the user query.

Therefore, the IR problem is formulated as:

Definition of IR Problem

The key goal of any IR System is to retrieve all the documents that are relevant to a user query, while retrieving as few as non-relevant items as possible.

Architecture of an IR system



Formulating the query

- It begins with the user's information need which is framed into the natural language query.
- The IR system converts the natural language query into a system query that is more convenient for the IR System to process.

Steps in IR process : 1. Crawling

Once the system query has been created, the whole information retrieval process can be viewed as a collection of three processes.

1. Crawling
2. Indexing
3. Retrieval and Ranking

Crawling : Web is considered as a collection of documents. the purpose of a web crawler is to collect various documents and store then in a repository called *"Document Collection"*

Steps in IR process : 2. Indexing Process

Need for indexing : The actual text of the document is not used in the retrieval process. Instead, documents in a collection are represented through a set of index terms or keywords.

Definition of Indexing

The process of transforming document into representation using index terms is called indexing.

Inverted Index : There are different types of index structures. One example is called inverted index. An inverted index is a list of keywords with each keyword carrying pointers to documents that contain that keyword.

Steps in IR process : 2. Indexing Process

Inverted Index Example

ID	Text
1	Baseball is played during summer months.
2	Summer is the time for picnics here.
3	Months later we found out why.
4	Why is summer so hot here

↑ Sample document data

Dictionary and posting lists →

Term	Freq	Document ids
baseball	1	[1]
during	1	[1]
found	1	[3]
here	2	[2], [4]
hot	1	[4]
is	3	[1], [2], [4]
months	2	[1], [3]
summer	3	[1], [2], [4]
the	1	[2]
why	2	[3], [4]

lucid IMAGINATION

Steps in IR process : 2. Indexing Process

Reducing index term set : The two commonly used text operations to reduce the set of representative keywords are:

Stop Word Removal :
Stop words are high frequency words which have little semantic weight - unlikely to help in retirieval.

Elimintating these stop words reduces thge numbers of index terms.

Some of the stop words are - about, above, after, all, an, am, is, was, the, of, if, for, else, near, why, were

Stemming :
Stemming is the process of removing affixes from the words to reduce them to stem.

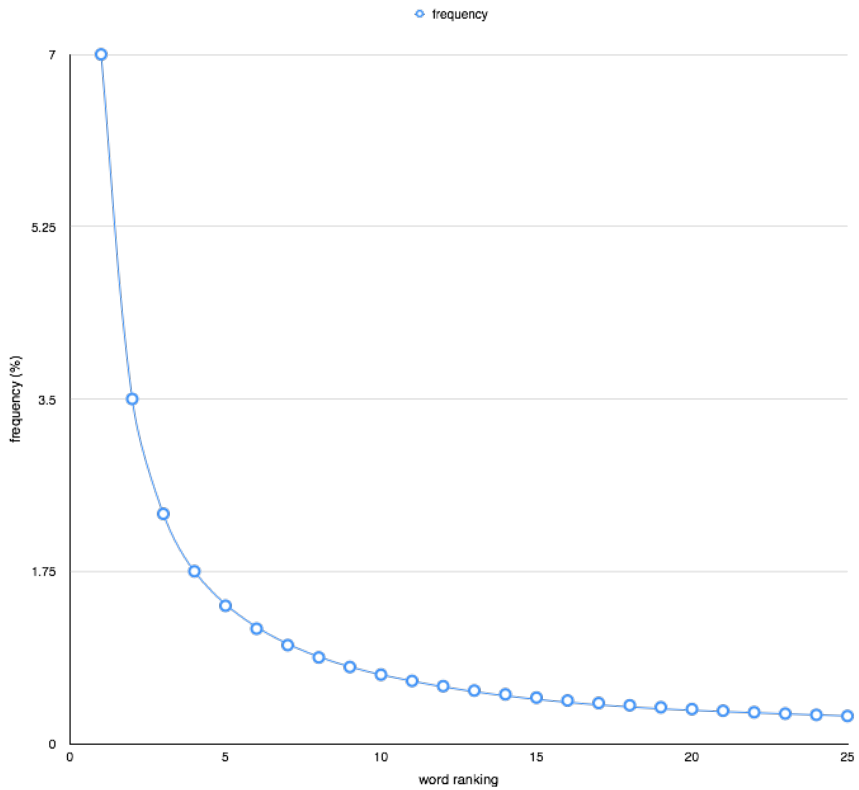
Thus the keywords are index terms in the form of stem and not the words as such.

{compute, computing, computes, computer}

$\longrightarrow$ *Stemming* $\longrightarrow$ "*comput*"

Steps in IR process : 2. Indexing Process - Zipf's law

Zipf's law states that the frequency of words multiplied by their ranks in a large corpus is more or less constant.
This indicates that the frequency of a word is inversely proportional to its rank .



Steps in IR process : 3. Retrieval and Ranking

Retrieval

The information retrieval system retrieves the documents that seem relevant to the query. The retrieval process is performed by matching the query representation with the document representation.

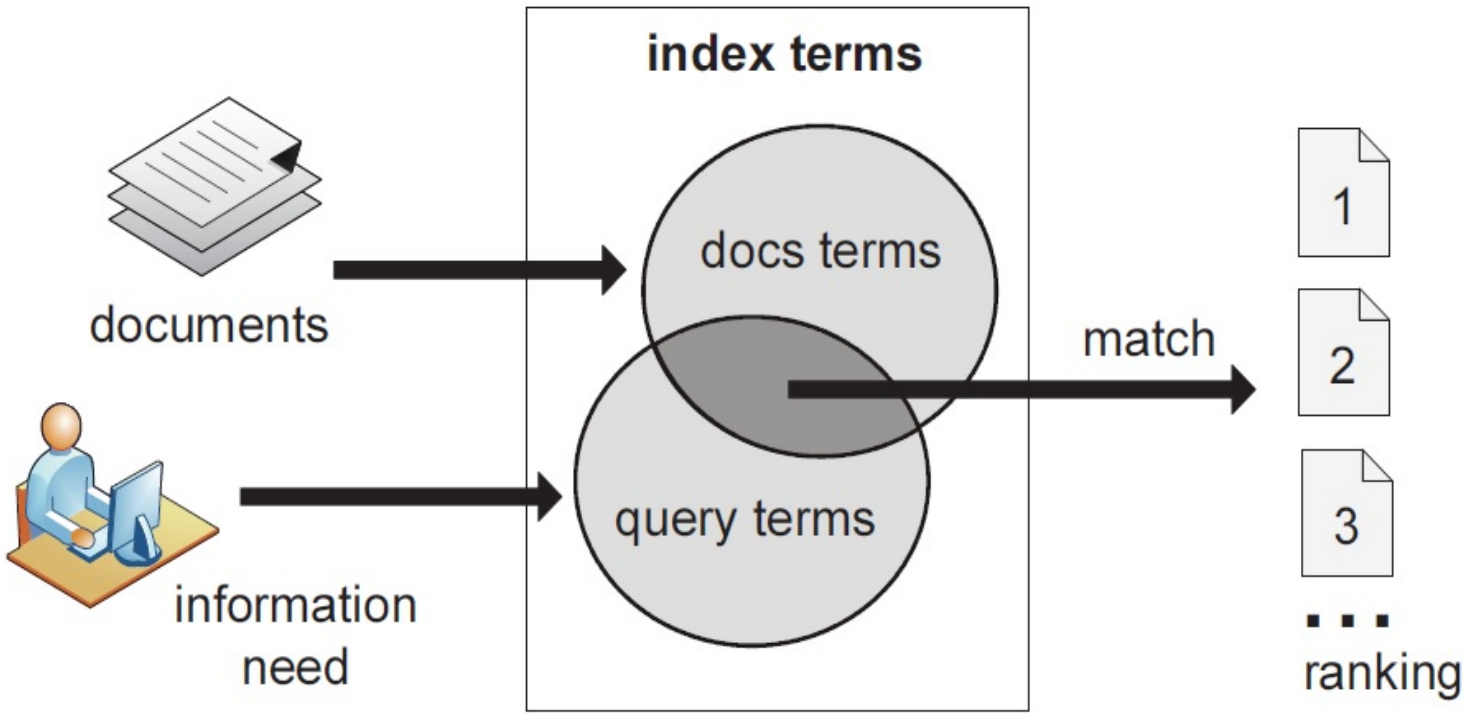
Ranking

Given a query q and a collection of documents D that match the query, the problem is to rank the documents in D according to some criteria, so that the "best" results appear early in the list displayed to the user.

What is IR Model?

An IR model is a pattern that defines several aspects of retrieval procedure. It represents how:

- documents and user queries are represented.
- an IR system retrieves matching documents.
- relevant documents are ranked.



What is IR Model?

Formal definition of IR model

An IR Model is a quadruple

$$[D, Q, F, R(q_i, d_j)]$$

where

1. D is a set of documents in the collection.
2. Q is a set of user queries
3. F is a framework for modelling documents and queries
4. $$R(q_i, d_j)$$

   is a ranking function

Types of IR Models

The types of IR models are defined as follows:

1. Classical IR Models

Classical IR models are based on mathematical knowledge. These models are simple, efficient and easy to implement.

2. Non - classical IR Models

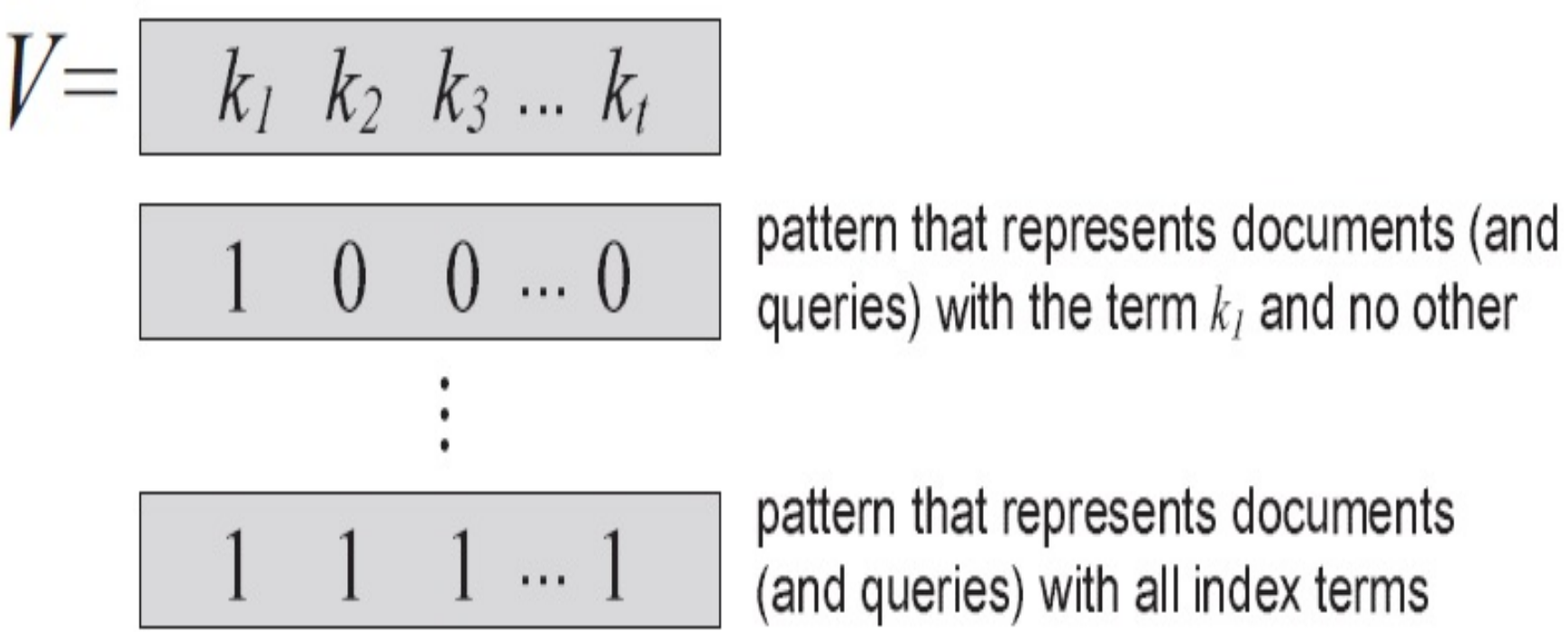
Non - classical IR models are exemplified by special logic techinique, situation theory and the concept of interaction.

3. Alternative IR Models

Alternative IR models are actually enhancements of classical IR models. They use techniques like cluster models, fuzzy model and latent semantic indexing.

Some Basic Concepts in IR

1. **1. Vocabulary :** V = {K1, K2, . . . , Kt} is the set of all distinct terms in the collection.
2. **2. Term Conjunctive Component :** Documents and queries can be represented by patterns of terms-co-coourences.



Some Basic Concepts in IR

3. **Term Document matrix :**

- 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

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

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

Types of Classical IR Models

1. Boolean Model
2. Term Weighting
3. Vector Space Model
4. Probabilistic Model

Boolean Model

- Boolean Model is the oldest of the classical IR models. It is based on Boolean Logic and Set theory.
- In this model, the documents are represented as a set of keywords with inverted indexing.
- Users are expected to express their **queries** as **Boolean expressions** consisting of keywords connected with Boolean logical operators.

Definition of Boolean Model

- T is a finite set of index terms. {t1, t2, ... , tm}.
- D is a finite set of documents. {d1, d2, ... , dn}.
- The query is usually a Boolean expression in Normal Form.

$$Q = \wedge(\vee\theta_i) \quad , \theta_i \epsilon \{t_i, \neg t_i\}$$

Boolean Model

Retrieval in Boolean Model

The retrieval is a two step process.

1. Obtain the set Si of documents that contain or do not contain the term ti.

$$Si = \{ d_j \mid \theta_i \epsilon d_j \}, \ \theta_i \epsilon \{ t_i , \neg t_i \}$$

2. Set Operations are used to retrieve documents in response to $Q = \frown Si$.

Boolean Model - Example

Let the set of documents be D = { D1, D2, D3 }.
where
D1 = Information retrieval is concerned with relevance to the query.
D2 = User's information is formulated into an user query.
D3 = Efficiency of retrieval depends on relevance to the user query.

The index terms set T = { information, retrieval, query }

User gives the query as , Q = information $\wedge$ *retrieval*

Boolean Model - Example

Step 1: Index the documents for keywords
d1 = { information, retrieval, query }
d2 = { information, query }
d3 = { retrieval, query }

Step 2 : Find the set of documents containing the term ti. As the query is Q = information $\wedge$ *retrieval*,

$$S_1 = \{ d_j \mid information \in d_j \} = \{ d1 , d2 \}$$

$$S_2 = \{ d_j \mid retrieval \in d_j \} = \{ d1 , d3 \}$$

Step 3 : The set operation is AND as inferred from the query.

So, Ans = S1 $\frown$ $S2$

$= \{ d1, d2 \} \frown \{ d1, d3 \}$

$= \{d1\}$

Pros :

1. Simple
2. Efficient
3. Easy to implement
4. Nice score for "recall" and "precision" if the query is well formulated.

Cons :

- The model is not able to retrieve documents that are **"Partial Matching"**
- The Boolean Model is **not able to rank** the retrieved documents.
- We cannot expect the user to formulate the user query into pure Boolean expressions.

Term Weighting Model

Each term that is selected as an indexing feature for documents is given a weight in this model.

Definition of Term Weighting Model

Let

- Ki be an index term and dj be the document.
- V = { K1, K2, ... , Kt } be th set of all index terms.
- The weight associated with (Ki,dj) is

$$w_{i,j} \geq 0$$

Basic Weighting scheme : The weights w i,j can be computed using the frequencies of occurrences of the term within documents.

1. The frequency of occurrence of the index term Li in the document dj is given by :

$$f_{i,j}$$

2. The total frequency of occurrence Fi of the index term Ki is

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

3. Document Frequency ni of the term Ki is the number of documents which contain the index term Ki. Note that ni $\leq Fi$

Sample Collection of documents

Document 1

To do is to be. To be is to do.

Document 2

To be or not to be. I am what i am.

Document 3

I think therefore i am. Do be do be do.

Document 4

Do do do, da da da . Let it be, let it be.

Step 0 : Figure out the important index terms : V = { to, be, do , i, am}

Step 1: Computing f values

Index Terms	**Doc1**	**Doc2**	**Doc3**	**Doc4**	**ni**	**Fi**
to	f(to,d1)=4	f(to,d2)=2	f(to,d3)=0	f(to,d4)=0	2	6
be	2	2	2	2	4	8
do	2	0	3	3	3	8
I	0	2	2	0	2	4
am	0	2	1	0	2	3

Table 1

TF-IDF term weighting scheme involves two important measures:

1. Term Frequency (TF)
2. Inverse Document Frequency (IDF)

TF : Term frequency was defined by Luhn as the frequency of occurrence of the term in the document. It is denoted by $tf_{i,j}$.

$$tf_{i,j} = \begin{cases} 1 + \log_2 f_{i,j} & f_{i,j} > 0 \\ 0 & otherwise \end{cases}$$

Step 2: Computing TF values

Index Terms	**tf i,1**	**tf i,2**	**tf i,3**	**tf i,4**
to	3	2	0	0
be	2	2	2	2
do	2	0	2.585	2.585
I	0	2	2	0
am	0	2	1	0

Table 2

Note that

$$\log_2 x = \frac{\log_{10} x}{log2} = \frac{\log_{10} x}{0.301}$$

The main problem with the TF score is the **"Document exhaustivity"** - the more index terms are assigned to a document, the higher is the probability of retrieval for that document.
We know that the terms are distributed in a text according to Zipf's law.

$$n(r)\ \alpha\ r^{-\alpha}$$

$$n(r)\ =\ Cr^{-\alpha}$$

Setting $\alpha = 1$ *for English vocabulary* , *we get*, $n(r)\ = \frac{C}{r}$

$$log\ n(r)\ = log\ c - log\ r$$

$$log\ r\ = log\ c - log\ n(r)$$

. Hence we arrive at the formula for inverse document frequency score.

Formula for IDF Score

$$IDF_i = \log_2 \frac{N}{n_i}$$

Step 3: Computing IDF scores

Index Terms	**ni**	**IDFi = log (N / ni)**
to	2	1
be	4	0
do	3	0.415
I	2	1
am	2	1

Table 3

Step 4 : Computing weights of each term

WIth repesct to the TF-IDF weighting scheme, we need to combine the TF and IDF scores.
The weight associated with the term Ki and the document dj is denoted by

$$w_{i,j}$$

Formula for $w_{i,j}$

$$w_{i,j} = \begin{cases} (1 + \log_2 f_{i,j}) \times \log_2 \frac{N}{n_i} & f_{i,j} > 0 \\ 0 & otherwise \end{cases}$$

So $w_{i,j}$ *is computed for our index term set as follows* :

Step 4 : Computing weights of each term

Index Terms	tf i,1	tf i,2	tf i,3	tf i,4	IDF-i	w i,1	w i,2	w i,3	w i,4
to	3	2	0	0	1	3	2	0	0
be	2	2	2	2	0	0	0	0	0
do	2	0	2.585	2.585	0.415	0.83	0	1.073	1.073
I	0	2	2	0	1	0	2	2	0
am	0	2	1	0	1	0	2	1	0

Table 4

Variants of TF-IDF Weighting scheme

Some variants have been suggested for tf score computing.

Scheme	TF formula
binary	$\{ 0 , 1 \}$
raw frequency	$f_{i,j}$
log normalisation	$1 + \log_2 f_{i,j}$
double normalisation	$0.5 + 0.5 \times \frac{f_{i,j}}{max\ f_{i,j}}$

Table 5

Pros :

1. Very easy to compute query - document and document - document similarity.
2. Easy to implement

Cons :

- Based on "bag of words" model
- Only useful as a lexical - level feature.

Vector Space Model

- The vector space model represents documents and queries as vectors of features representing the terms that occur with them.
- Each document is characterized as a Boolean or numerical vector.
- Each feature takes a value of either zero or one, indicating the absence or presence of that term in a document or query.

Definition of Vector Space Model

* T is a finite set of index terms. {t1, t2, ... , tm}.
* D is a finite set of documents. {d1, d2, ... , dn}.
* Each document is represented by a column vector of weights

$$\{w_{1j}, w_{2j}, w_{3j}, ..., w_{ij}, ..., w_{mj}\}^T$$

Features of Vector-Space Model

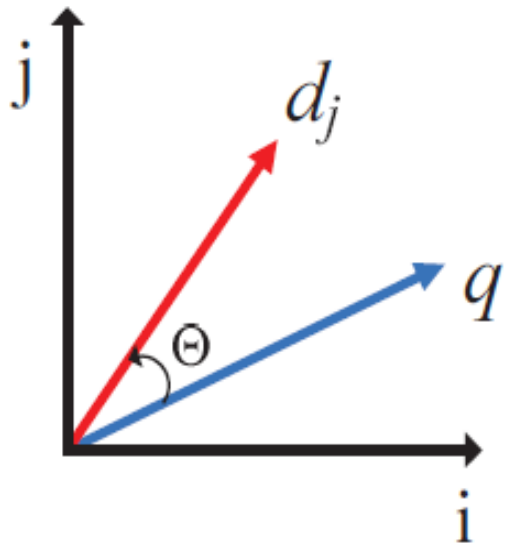
1. Weight associated with a pair (Ki,dj) is positive and non-binary.
2. The index terms are mutually independent.
3. The representations of document dj and query q are t-dimensional vectors.

$$\overrightarrow{d_j} = (w_{1j}, w_{2j}, ..., w_{tj})$$

$$\overrightarrow{q} = (w_{1q}, w_{2q}, ..., w_{tq})$$

Similarity Measure

- Similarity between a document d_j and a query q



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

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

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

Formulas page and Example Query

$$w_{ij} = (1 + \log_2 f_{i,j}) \times \log_2 \frac{N}{n_i}$$

$$w_{iq} = (1 + \log_2 f_{i,q}) \times \log_2 \frac{N}{n_i}$$

Consider the example query as : **"to do"**
So the formula becomes, when considering the doc 1 :

$$sim(d1, q) = \frac{\sum_{i=1}^{2} w_{1j} \times w_{1q}}{\sqrt{\sum_{i=1}^{2} w_{1j}}} \quad (1)$$

$$= \frac{\{Considering\ "to"\}[w_{to,1} \times w_{to,q}] + \{Considering\ "do"\}[w_{do,1} \times w_{do,q}]}{Denominator = \ called\ as\ Document\ length\ normalisation}$$

Computing w_{iq} values

Index Terms	$\mathbf{f}_{iq}$	$\mathbf{tf}_{iq}$	$\mathbf{IDF}_i$	$\mathbf{w}_{iq}$
to	1	$1+\log_2 1 = 1+0 = 1$	1	1
do	1	$= 1 + 0 = 1$	0.415	0.415

Table 6 : Computing w iq values

$$w_{to,q} = 1 \tag{2}$$

$$w_{do,q} = 0.415 \tag{3}$$

Derivation of similarity score between document and query

Document Number	$w_{to,j}$	$w_{to,q}$	$w_{do,j}$	$w_{do,q}$	**Similarity measure numerator**
Doc1	3	1	0.83	0.415	$=3\times1+0.83\times0.415$ $=4.245$
Doc2	2	1	0	0.415	$=2\times1+0\times0.415$ $=2$
Doc3	0	1	1.073	0.415	$=0\times1+1.073\times0.415$ $=0.445$
Doc4	0	1	1.073	0.415	$=0\times1+1.073\times0.415$ $=0.445$

Table 7 : Numerator in similarity score

Tips to help in Table 7

-> Column 3 (w to,q) : is populated using equation 2

-> Column 5 (w do,q) : is populated using equation 3

-> Column 2 (w to,j) and Column 4 (w do,j): are populated from table 4.

-> Column 6 in each row = Col 2 * Col 3 + Col 4 * Col 5

Till now, we have found the numerator value in equation 1. To find denominator, find "Document Length Normalisation".

Document length Normalisation : Denominator in eq 1

We need to find the document length of each document .

$\overrightarrow{d1} = \sqrt{\sum {w_{i1}}^2} = \sqrt{(w_{to,1})^2 + (w_{do,1})^2 + (w_{be,1})^2 + (w_{I,1})^2 + (w_{am,1})^2}$

Refer table 4 for these values

$= \sqrt{3^2 + 0.83^2} = 3.112$

Similarly for other documents,

$\overrightarrow{d2} = \sqrt{2^2 + 0^2 + 0^2 + 2^2 + 2^2} = \sqrt{12} = 3.464$

$\overrightarrow{d3} = \sqrt{1.073^2 + 2^2 + 2^2} = \sqrt{6.1513} = 2.4801$

$\overrightarrow{d4} = \sqrt{1.073^2} = 1.073$

Ranking in Vector Space Model

Docs	**Numerator (from Tab 7)**	**Denominator (from prev slide)**	**Similarity score** (Col2/Col3)	**Rank**
Doc1	4.245	3.112	1.364	1
Doc2	2	3.464	0.577	2
Doc3	0.445	2.4801	0.1794	4
Doc4	0.445	1.073	0.414	3

Table 8 : Ranking in vector space model

Pros and Cons

Pros :

1. Term - weighting improves quality of the answer set.
2. Cosine similarity is more efficient.

Cons : Assumes independence of index terms

Probabilistic Model

- The probabilistic model applies a probabilistic framework to IR.
- It ranks documents based on the probability of their relevance to a given query.
- Given a user query, there is an **ideal answer set** for this query.
- The IR uses the initial set of documents retrieved to refine the description of the ideal answer set.

Ranking in probabilistic model

- Let,
  - R be the set of relevant documents to query q
  - $\overline{R}$ be the set of non-relevant documents to query q
  - $P(R|\vec{d}_j)$ be the probability that d_j is relevant to the query q
  - $P(\overline{R}|\vec{d}_j)$ be the probability that d_j is non-relevant to q
- The similarity $sim(d_j, q)$ can be defined as

$$sim(d_j, q) = \frac{P(R|\vec{d}_j)}{P(\overline{R}|\vec{d}_j)}$$

Derivation for Similarity Score

After applying Bayes theorem and using logarithm operations, we get

- Further, assuming that
  - $\forall\, k_i \notin q,\quad p_{iR} = q_{iR}$

  and converting the log products into sums of logs, we finally obtain

$$sim(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log\left(\frac{p_{iR}}{1-p_{iR}}\right) + \log\left(\frac{1-q_{iR}}{q_{iR}}\right)$$

Term Incidence Contingency table

- Let,
  - N be the number of documents in the collection
  - n_i be the number of documents that contain term k_i
  - R be the total number of relevant documents to query q
  - r_i be the number of relevant documents that contain term k_i
- Based on these variables, we can build the following contingency table

	relevant	non-relevant	all docs
docs that contain k_i	r_i	$n_i - r_i$	n_i
docs that do not contain k_i	$R - r_i$	$N - n_i - (R - r_i)$	$N - n_i$
all docs	R	$N - R$	N

Derivation of Similarity Score

- If information on the contingency table were available for a given query, we could write
  - $p_{iR} = \frac{r_i}{R}$
  - $q_{iR} = \frac{n_i - r_i}{N - R}$
- Then, the equation for ranking computation in the probabilistic model could be rewritten as

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

where $k_i[q, d_j]$ is a short notation for $k_i \in q \wedge k_i \in d_j$

Derivation of Similarity Score

For handling small values of r_i, we add 0.5 to each of the terms in the formula above, which changes $sim(d_j, q)$ into

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

This formula is considered as the classic ranking equation for the probabilistic model and is known as the Robertson-Sparck Jones Equation

Derivation of Similarity Score

The previous equation cannot be computed without estimates of r_i and R

One possibility is to assume $R = r_i = 0$, as a way to boostrap the ranking equation, which leads to

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

Derivation of Similarity Score

The ranking computation led to negative weights because of the term "do"

Actually, the probabilistic ranking equation produces negative terms whenever $n_i > N/2$

One possible artifact to contain the effect of negative weights is to change the previous equation to:

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

Example Problem in Probabilistic Model

Consider the same document collection. The query given to the IR system is **"to do"**.

Step 1 : Find the ni value of each term in the query.

$$n_{to} = 2$$

$$n_{do} = 3$$

Note that N = 4.

Step 2 : Find the score of the "**to**" term

$$\frac{\lg(4 + 0.5)}{\lg(2 + 0.5)} = 0.847$$

Step 3 : Find the score of the "**do**" term

$$\frac{\lg(4 + 0.5)}{\lg(3 + 0.5)} = 0.362$$

Similarity Score Computation

Docs	**Contains "to" ?**	**Contains "do" ?**	**Similarity score computation**	**Rank**
Doc1	✓	✓	$0.847 + 0.362 = 1.21$	1
Doc2	✓		0.847	2
Doc3		✓	0.362	3
Doc4		✓	0.362	3

Table 9 : Ranking in probabilistic model

Pros and Cons

Pros : Documents are ranked in decreasing order of probability of relevance

Cons :

- Initial answer set is required.
- This method does not consider the TF values.
- It suffers from lack of document length normalization.

Non-Classical IR Models

While classical IR models are based on similarity and probability, the non-classical IR models are based on the three theories:

1. Information logic Model
2. Situation theory model
3. Interaction model

Information Logic Model

- Information logical model is based on **"logical imaging"**.
- Retrieval is performed by making inferences from document to query.
- A measure of uncertainty is associated with the inference based on Rijsbergen's Principle .

Rijsbergen's Principle

Given any two sentences x and y, a measure of the uncertainity of

$$y \rightarrow x$$

relative to a given data set is determined by the minimal extent to which one has to add information to the data set in order to establish the truth

$$y \rightarrow x.$$

Information logic model was developed because "Classical models were unable to enhance effectiveness. "

Situation Theory Model

- ⇒ Situation Theory is also based on Rijsbergen's principle.
- ⇒ Retrieval is considered as a flow of information from document to query.
- ⇒ A structure called **"infon"** is used to describe the situation and to model information flow .
- ⇒ An infon represents a n-ary relation and its polarity.
- ⇒ The polarity of an infon can be either 1 or 0, representing that the infon carries out positive or negative information respectively.

Example of an infon

Situation : I see Vijay serving the dish

Corresponding infon is $\Rightarrow \imath = << Serving\ \ Vijay,\ \ dish\ ; 1 >>$

Support of an infon : Support of an infon is represented as

$$S \mid = \imath$$

meaning that "situation s makes the infon $\imath$ *is true*"

Relevance to query :
A document d is relevant to a query q, if

$$d \mid = \ q$$

. Also, if does not support q, it is not that document is not relevant to the query. Additional information such as synonyms, hypernyms/hyponyms , meronymns can be used to transform the document d to d$^{'}$ *such that* $d^{'} \mid = \ q$
This transformation is called **flow of information between situations.**

Interaction Model

- First introduced in Dominich and Rijsbergen.
- In this model, the documents are not isolated. Instead, they are interconnected.
- The query interacts with the interconnected documents.
- Retrieval is considered as a result of this interaction.

Artificial neural networks approach :

- ANN can be used to implement this interaction model.
- Each document is modelled as a neuron, the document set as a whole forms a neural network.
- The query is also modelled as a neuron and integrated into the network.
- Because of this integration, new connections are built between the query and the documents and existing connections are changed.
- This restructuring corresponds to the concept of interaction.

Interaction Model

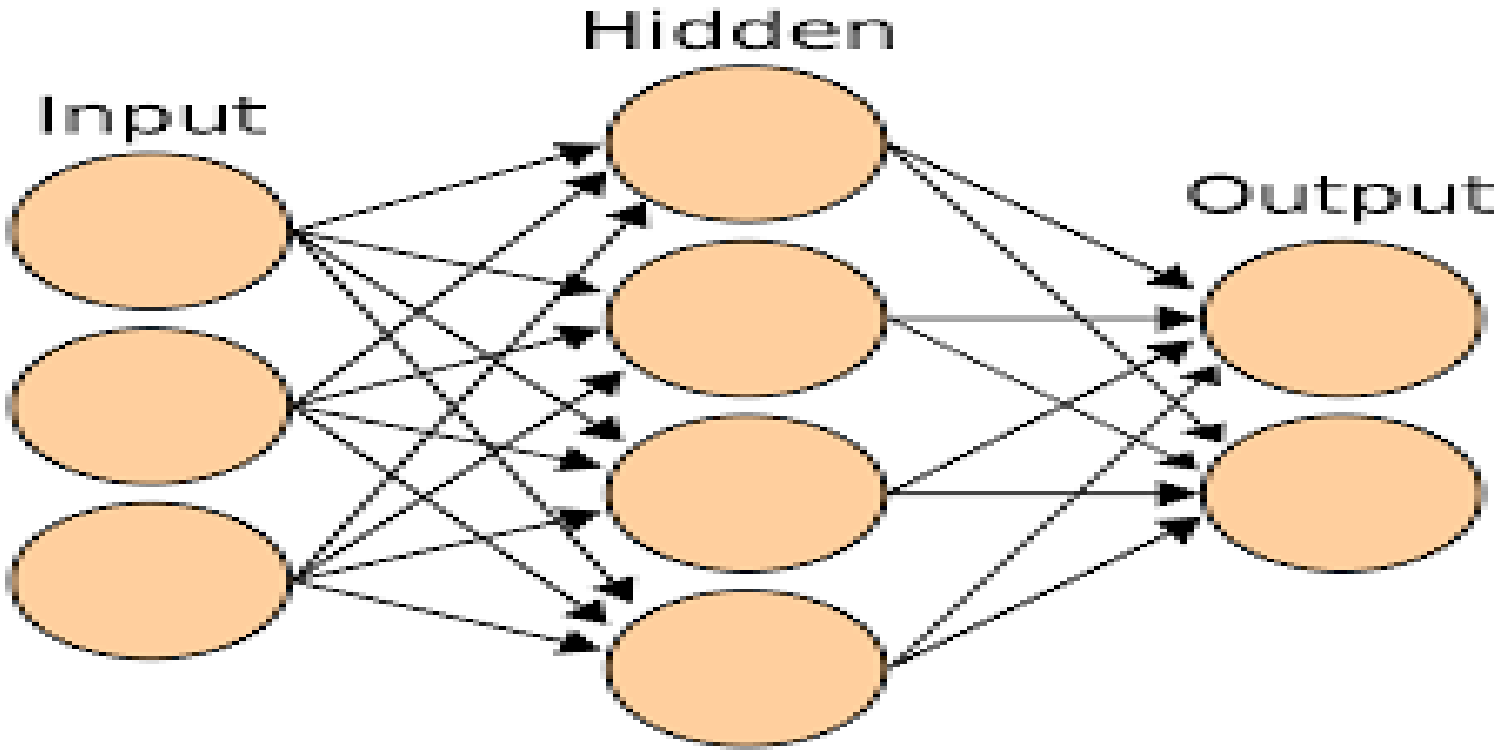


Figure: Artificial Neural network of documents and queries

Alternative Model 1 : Fuzzy Model

- In fuzzy model, the document is represented as a fuzzy set of terms
  [ti,$\mu(ti)$] *where* μ *is the membership function.*

- The membership degree expresses the significance (weights) of term to the information contained in the documents
- The membership function assigns a membership degree to each term of the document.

Alternative Model 1 : Fuzzy Model

- Each term ti is represented by a fuzzy set fi
- Fuzzy set operators are applied to obtain the desired result
- For single term query q=tq, documents from the fuzzy set fq are retrieved
- For AND query q=tq1 ^ tq2
  - Fuzzy sets fq1 and fq2 are obtained
  - Fuzzy intersection operator is used to obtain the resultant set

    fq1 ^ fq2 = min{(dj,wq1), (dj,wq2)}
  - The documents in this set are returned
- For OR query q=tq1 V tq2
  - Fuzzy sets fq1 and fq2 are obtained
  - Fuzzy intersection operator is used to obtain the resultant set

    fq1 V fq2 = max{(dj,wq1), (dj,wq2)}
  - The documents in this set are returned

Alternative Model 1 : Fuzzy Model

- D1={information, retrieval, query}
- D2={retrieval, query, model}
- D3={information, retrieval}

- Vocabulary = {information, model, query, retrieval}

- The fuzzy sets induced by these terms are

  f1={(d1, 1/3), (d2,0), (d3, ½)}
  f2={(d1, 0), (d2,1/3), (d3, 0)}
  f3={(d1, 1/3), (d2,1/3), (d3, 0)}
  f4={(d1, 1/3), (d2,1/3), (d3, ½)}

Try : q= t1 v t4

- Query q= t2 ^ t4 (model retrieval)
  - Fuzzy sets f2 and f4 are considered
  - Min(f2(d1), f4(d1)), Min(f2(d2), f4(d2)), Min(f2(d3), f4(d3))
  - = {(d1, 0), (d2,1/3), (d3, 0)}
  - d2 is returned

Alternative Model 2 : Cluster Model

- This model is an attempt to reduce the number of matches during retrieval.
- Hypothesis: Closely associated documents tend to be relevant to the same clusters.
- Instead of matching the query with every documents in the collection, it is matched with representatives of the class.

Alternative Model 2 : Cluster Model

- Let D={d1,d2,...,dm} – finite set of documents
- Let E=(eij)n,n – similarity matrix (or distance matrix)
- Let T be the threshold
- Any pair of documents di and dj (i ≠ j) whose similarity measures exceeds T or whose distance is less than T is grouped to form a cluster
- The remaining document form a single cluster.
- A representative vector of each class is constructed by computing the centroid of the document vector (mean)
- During retrieval, the query is compared with the cluster vectors based on similarity or distance

Alternative Model 2 : Cluster Model

- Let term-document matrix

	d1	d2	d3
A =	1	1	0
	0	1	0
	1	1	1
	0	0	1
	1	1	0

- The similarity matrix

	d1	d2	d3
d1	1.0		
d2	0.9	1.0	
d3	0.4	0.4	1.0

Alternative Model 2 : Cluster Model

- Threshold T = 0.7
- We get two clusters

  C1 ={d1,d2} C2={d3}
- The cluster vectors (representatives) for C1 and C2

  r1=(1 0.5 1 0 1)

  r2=(0 0 1 1 0)
- Retrieval is performed by matching the query vector with r1 and r2

Alternative Model 3 : Latent Semantic Indexing Model

- LSI attempts to identify the hidden semantic structure through statistical techniques and use it to represent and retrieve information.
- This is done by modelling the association between terms and documents based on **co-occurrence**.
- LSI transforms the term-document vector space into a more compact latent semantic space.
- In the latent semantic space, a query and a document can have high similarity even if the document does not contain a query term, provided the terms are semantically related.

LSI model - Definition

Definition of LSI model : Term - Document Matrix

- ★ The document collection is processed to get m X n term-document matrix W , where
  - m = number of index terms
  - n= total number of documents in the collection
- ★ **Columns** of this matrix represent **document vectors** while **rows** represent **term vectors**.
- ★ Matrix element
  W_{ij} *represents the weight of the term i in the document j .*

LSI model - Definition

Definition of LSI model : Singular Value Decomposition

Singular Value Decomposition (SVD) of the term-document matrix is then computed.

- Using SVD, the matrix W is represented as a product of three matrices

$$W = TSD^T$$

- T corresponds to term vectors.
- T has m rows and min(m,n) columns
- S corresponds to singular values.

The transpose of D (Document Vector] is called D^T

Evaluation of the IR System

Evaluation of the IR System is the process of assessing how well a system meets the information needs of its users.

The types of IR evaluation models are :

1. **System driven models :** measure how well a system ranks documents.
2. **User centered models :** measure user satisfaction.

Cleverdon listed the following six criteria for evaluation.

1. Coverage of the collection
2. Time lag
3. Presentation format
4. User effort
5. Precision
6. Recall

1. Relevance

- Relevance is **subjective** in nature - it depends on the individual judgements of the user.
- Given a query, the same document may be judged as relevant by one user and non-relevant to another user.
- So it is not possible to measure "true relevance" .
- Most evaluation of IR system have so far been done on test document collections with known relevance judgements.

Problem with relevance : Another issue with relevance is the **degree of relevance**. Relevance has been visualised a binary concept - either relevant or non-relevant.

2. Effectiveness Measure

Effectiveness is purely a measure of the ability of the system to satisfy the user in terms of relevance of documents retrieved.
The two most commonly used effectiveness measures are - Precision and recall.

Precision:

1. Defined as the proportion of the retrieved docuemnts that are relevant.
2. Measures the accuracy of the system

Recall :

- The proportion of the relevant documents that are retrieved.
- Measures its exhaustiveness.

Precision and recall formulae

$$Precision = \frac{Number\ of\ relevant\ documents\ retrieved\ NR_{ret}}{Total\ number\ of\ documents\ retrieved\ N_{ret}}$$

$$Recall = \frac{Number\ of\ relevant\ documents\ retrieved\ NR_{r}}{Total\ number\ of\ relevant\ documents\ in\ the\ collection\ retrievedNR}$$

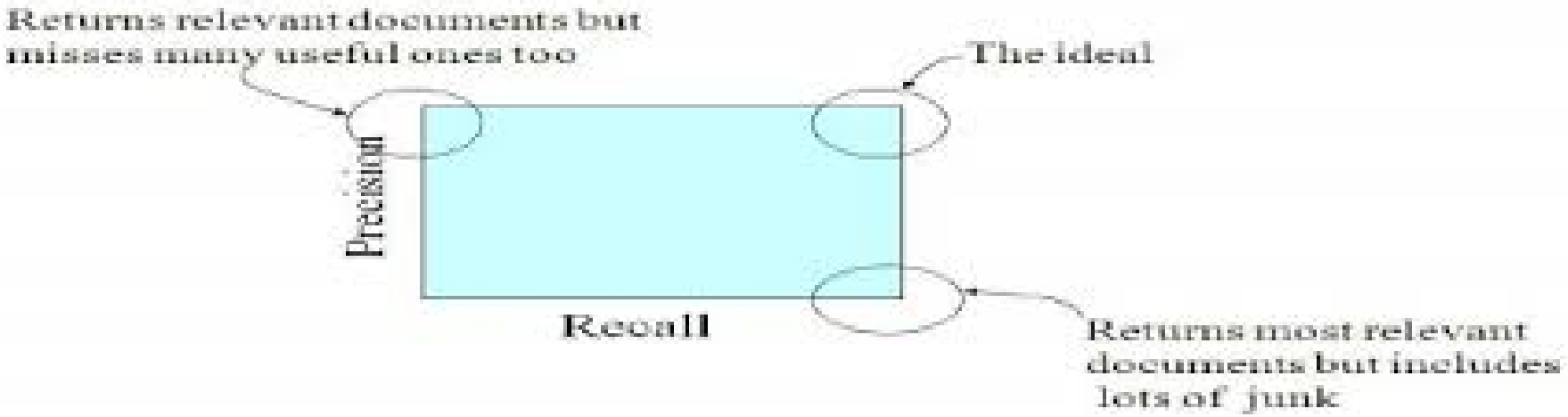


Figure: Trade off between precision and recall