Chapter 3
Modeling

Introduction to IR Models
Basic Concepts
The Boolean Model
Term Weighting
The Vector Model
Probabilistic Model
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
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 and it is simple to refer to in a query

Simplicity is important because it reduces the effort of query formulation
Introduction

Information retrieval process

documents

information need

index terms

docs terms

query terms

match

1

2

3

... ranking
Introduction

- A ranking is an ordering of documents that (hopefully) reflects the relevance of the documents to a user query.
- Thus, any IR system has to deal with the problem of predicting which documents the users will find relevant.
- This problem naturally embodies a degree of uncertainty, or vagueness.
An IR model is a quadruple \([D, Q, \mathcal{F}, R(q_i, d_j)]\) where

1. \(D\) is a set of logical views for the documents in the collection
2. \(Q\) is a set of logical views for the user queries
3. \(\mathcal{F}\) is a framework for modeling document representations, queries, and their relationships
4. \(R(q_i, d_j)\) is a ranking function which associates a real number with a query \(q_i \in Q\) and a document \(d_j \in D\).
IR Models

A taxonomy of information retrieval models
Retrieval: Ad Hoc x Filtering

Ad Hoc Retrieval:

Collection
Fixed size

Q1
Q2
Q3
Q4
Filtering

Classic 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,
- \( t \) the number of index terms in the document collection
- \( k_i \) a generic index term

Then,
- The **vocabulary** \( V = \{k_1, \ldots, k_t\} \) is the set of all distinct index terms in the collection

\[
V = \begin{bmatrix} k_1 & k_2 & k_3 & \cdots & k_t \end{bmatrix}
\]

vocabulary of \( t \)
index terms
Basic Concepts

Documents and queries can be represented by patterns of term co-occurrence of terms

$$V = \begin{array}{cccc}
  & k_1 & k_2 & k_3 & \ldots & k_t \\
\hline
1 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & 1 & \ldots & 1 \\
\end{array}$$

- Pattern that represents documents (and queries) with the term $k_1$ and no other
- 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 occurrence of a term in a document establishes a relation between them

A term-document relation can be quantified by the frequency of the term in the document

In matrix form, this can be written as

\[
\begin{bmatrix}
k_1 & f_{1,1} & f_{1,2} \\
k_2 & f_{2,1} & f_{2,2} \\
k_3 & f_{3,1} & f_{3,2}
\end{bmatrix}
\]

where each \( f_{i,j} \) element stands for the frequency of term \( k_i \) in document \( d_j \)
Logical view of a document: from full text to a set of index terms
Classic IR Models
The Boolean Model
The Boolean Model

Simple model based on set theory and boolean algebra

Queries specified as boolean expressions

- quite intuitive and precise semantics
- neat formalism
- example of query: \( q = k_a \land (k_b \lor \neg k_c) \)

Term-document frequencies in the term-document matrix are all binary

- \( w_{iq} \): weight associated with pair \((k_i, q)\)
- \( w_{iq} \in \{0, 1\} \): terms either present or absent
- \( \vec{d}_q = (w_{1q}, w_{2q}, ..., w_{tq}) \): weighted vector associated with \( q \)
The Boolean Model

A term conjunctive component that satisfies a query $q$ is called a **query conjunctive component** $c(q)$.

A query $q$ rewritten as a disjunction of those components is called the **disjunct normal form** $q_{DNF}$.

To illustrate, consider

- query $q = k_a \land (k_b \lor \neg k_c)$
- vocabulary $V = \{k_a, k_b, k_c\}$

Then

- $q_{DNF} = (1, 1, 1) \lor (1, 1, 0) \lor (1, 0, 0)$
- $c(q)$: a conjunctive component for $q$
The Boolean Model

The three conjunctive components for the query

\[ q = k_a \land (k_b \lor \neg k_c) \]
The Boolean Model

This approach works even if the vocabulary of the collection includes terms not in the query.

Consider that the vocabulary is given by
\[ V = \{k_a, k_b, k_c, k_d\} \]

Then, a document \(d_j\) that contains only terms \(k_a, k_b,\) and \(k_c\) is represented by \(c(d_j) = (1, 1, 1, 0)\)

The query \([q = k_a \land (k_b \lor \neg k_c)]\) is represented in disjunctive normal form as

\[ q_{DNF} = (1, 1, 1, 0) \lor (1, 1, 1, 1) \lor (1, 1, 0, 0) \lor (1, 1, 0, 1) \lor (1, 0, 0, 0) \lor (1, 0, 0, 1) \]
The Boolean Model

The similarity of the document $d_j$ to the query $q$ is defined as

$$sim(d_j, q) = \begin{cases} 1 & \text{if } \exists c(q) \mid c(q) = c(d_j) \\ 0 & \text{otherwise} \end{cases}$$

The Boolean model predicts that each document is either relevant or non-relevant.
Drawbacks of the Boolean Model

- Retrieval based on binary decision criteria with no notion of partial matching
- No ranking of the documents is provided (absence of a grading scale)
- Information need has to be translated into a Boolean expression, which most users find awkward
- The Boolean queries formulated by the users are most often too simplistic
- The model frequently returns either too few or too many documents in response to a user query
Classic IR Models
Term Weighting
The terms of a document are not equally useful for describing the document contents.

In fact, there are index terms which are simply vaguer than others.

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.
Term Weighting

To characterize term importance, we associate a weight $w_{i,j} > 0$ for 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 $d_j$ document.

These weights are useful to compute a rank for each document in the collection with regard to a given query.
Let,

- $k_i$ an index term and $d_j$ a document
- $V = \{k_1, k_2, ..., k_t\}$ the set of all index terms
- $w_{i,j} \geq 0$ the weight associated with $(k_i, d_j)$

Then we define $\vec{d}_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$ as a weighted vector that contains the weight $w_{i,j}$ of each term $k_i \in V$ in the document $d_j$.

**Diagram:**

- $V$ represents the vocabulary of $t$ index terms.
- $d_j$ represents the term weights associated with $d_j$. 
- Each term $k_i$ is connected to its weight $w_{i,j}$.
Term Weighting

The weights $w_{i,j}$ are computed based on 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 $d_j$.

Then we define the total frequency of occurrence $F_i$ of term $k_i$ in the collection 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** $n_i$ of a term $k_i$ is the number of documents in which it occurs.

Notice that $n_i \leq F_i$.

For instance, in the document collection below, the values $f_{i,j}$, $F_i$ and $n_i$ associated to the term *do* are:

- $f(\text{do}, d_1) = 2$
- $f(\text{do}, d_2) = 0$
- $f(\text{do}, d_3) = 3$
- $f(\text{do}, d_4) = 3$
- $F(\text{do}) = 8$
- $n(\text{do}) = 3$

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

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

Do do do, da da da. Let it be, let it be.
Term-term correlation matrix

For classic information retrieval models, the index term weights are assumed to be **mutually independent**

This means that $w_{i,j}$ tells us nothing about $w_{i+1,j}$

This is clearly a simplification because occurrences of index terms in a document are not uncorrelated

For instance, the terms **computer** and **network** can be used to index a document on computer networks

In this document, the appearance of one of these terms attracts the appearance of the other

Thus, they are correlated and their weights should reflect this correlation.
To take into account term-term correlations, we can compute a correlations matrix.

Let \( \tilde{M} = (m_{ij}) \) be a term-document matrix \( t \times N \) where \( m_{ij} = w_{i,j} \).

The matrix \( \tilde{C} = \tilde{M} \tilde{M}^t \) is a term-term correlation matrix.

Each element \( c_{u,v} \in C \) expresses a correlation between terms \( k_u \) and \( k_v \), given by

\[
c_{u,v} = \sum_{d_j} w_{u,j} \times w_{v,j}
\]

Higher the number of documents in which the terms \( k_u \) and \( k_v \) co-occur, stronger is this correlation.
Term-term correlation matrix

Term-term correlation matrix for a sample collection

\[
\begin{pmatrix}
  d_1 & d_2 \\
  k_1 & \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{pmatrix} \\
k_2 & \begin{pmatrix}
  k_1 \\
k_2 \\
k_3
\end{pmatrix}
\end{pmatrix}
\begin{pmatrix}
  k_1 \\
k_2 \\
k_3
\end{pmatrix}
\begin{pmatrix}
  d_1 \\
d_2
\end{pmatrix}
\]

\[
\begin{pmatrix}
  w_{1,1}w_{1,1} + w_{1,2}w_{1,2} & w_{1,1}w_{2,1} + w_{1,2}w_{2,2} & w_{1,1}w_{3,1} + w_{1,2}w_{3,2} \\
  w_{2,1}w_{1,1} + w_{2,2}w_{1,2} & w_{2,1}w_{2,1} + w_{2,2}w_{2,2} & w_{2,1}w_{3,1} + w_{2,2}w_{3,2} \\
  w_{3,1}w_{1,1} + w_{3,2}w_{1,2} & w_{3,1}w_{2,1} + w_{3,2}w_{2,2} & w_{3,1}w_{3,1} + w_{3,2}w_{3,2}
\end{pmatrix}
\]
Classic IR Models
TF-IDF Weights
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-term correlation matrix**

Luhn Assumption. The value of $w_{i,j}$ is proportional to the term frequency $f_{i,j}$

That is, the more often a term occurs in the text of the document, the higher its weight.

It is based on the observation that high frequency terms are important for describing documents.

This leads directly to the following $tf$ weight formulation:

$$tf_{i,j} = f_{i,j}$$
A variant of $tf$ weight used in the literature is

$$ t f_{i,j} = \begin{cases} 
1 + \log f_{i,j} & \text{if } f_{i,j} > 0 \\
0 & \text{otherwise}
\end{cases} $$

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.
**Term Frequency (TF) Weights**

Log $t_f$ weights $t_f_{i,j}$ for the example collection

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th>$t_{f_{i,1}}$</th>
<th>$t_{f_{i,2}}$</th>
<th>$t_{f_{i,3}}$</th>
<th>$t_{f_{i,4}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>to</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>do</td>
<td>2</td>
<td>-</td>
<td>2.585</td>
</tr>
<tr>
<td>3</td>
<td>is</td>
<td>2</td>
<td>2.585</td>
<td>2.585</td>
</tr>
<tr>
<td>4</td>
<td>be</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>or</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>not</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>I</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>am</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>what</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>think</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>therefore</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>da</td>
<td>-</td>
<td>-</td>
<td>2.585</td>
</tr>
<tr>
<td>13</td>
<td>let</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>it</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>
Inverse Document Frequency

We call **document exhaustivity** the number of index terms assigned to a document.

The more index terms are assigned to a document, the higher is the probability of retrieval for that document. If too many terms are assigned to a document, it will be retrieved by queries for which it is not relevant.

**Optimal exhaustivity.** We can circumvent this problem by optimizing the number of terms per document.

Another approach is by weighting the terms differently, by exploring the notion of **term specificity**.
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

- Terms are distributed in a text according to Zipf’s Law.
- Thus, if we sort the vocabulary terms in decreasing order of document frequencies we have

\[ n(r) \sim r^{-\alpha} \]

where \( n(r) \) refer to the \( r \)th largest document frequency and \( \alpha \) is an empirical constant.

- That is, the document frequency of term \( k_i \) is an exponential function of its rank.

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

where \( C \) is a second empirical constant.
Inverse Document Frequency

Setting $\alpha = 1$ (simple approximation for English collections) and taking logs we have

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

For $r = 1$, we have $C = n(1)$, i.e., the value of $C$ is the largest document frequency

This value works as a normalization constant

An alternative is to do the normalization assuming $C = N$, where $N$ is the number of docs in the collection

$$\log r \sim \log N - \log n(r)$$
Inverse Document Frequency

Let $k_i$ be the term with the rth largest document frequency, i.e., $n(r) = n_i$. Then,

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

where $idf_i$ is called the **inverse document frequency** of term $k_i$

Idf provides a foundation for modern term weighting schemes and is used by almost any IR system
**Inverse Document Frequency**

Idf values for example collection

<table>
<thead>
<tr>
<th>term</th>
<th>$n_i$</th>
<th>$idf_i = \log(N/n_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>do</td>
<td>3</td>
<td>0.415</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>be</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>or</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>not</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>l</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>am</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>what</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>think</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>therefore</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>da</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>let</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>it</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Term</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>do</td>
<td>0.830</td>
<td>2</td>
<td>1.073</td>
<td>1.073</td>
</tr>
<tr>
<td>is</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>be</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>or</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>not</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>am</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>what</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>think</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>therefore</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>da</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5.170</td>
</tr>
<tr>
<td>let</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>it</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>tf weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
</tr>
<tr>
<td>{0,1}</td>
</tr>
<tr>
<td>raw frequency</td>
</tr>
<tr>
<td>$f_{i,j}$</td>
</tr>
<tr>
<td>log normalization</td>
</tr>
<tr>
<td>$1 + \log f_{i,j}$</td>
</tr>
<tr>
<td>double normalization 0.5</td>
</tr>
<tr>
<td>$0.5 + 0.5 \frac{f_{i,j}}{\max_i f_{i,j}}$</td>
</tr>
<tr>
<td>double normalization K</td>
</tr>
<tr>
<td>$K + (1 - K) \frac{f_{i,j}}{\max_i f_{i,j}}$</td>
</tr>
</tbody>
</table>

Retrieval Evaluation, Modern Information Retrieval, Addison Wesley, 2006 – p. 45
## Variants of TF-IDF

Five distinct variants of idf weight

<table>
<thead>
<tr>
<th>Variant</th>
<th>idf weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>unary</td>
<td>1</td>
</tr>
<tr>
<td>inverse frequency</td>
<td>$\log \frac{N}{n_i}$</td>
</tr>
<tr>
<td>inv frequency smooth</td>
<td>$\log(1 + \frac{N}{n_i})$</td>
</tr>
<tr>
<td>inv frequency max</td>
<td>$\log(1 + \frac{\max_i n_i}{n_i})$</td>
</tr>
<tr>
<td>probabilistic inv frequency</td>
<td>$\log \frac{N-n_i}{n_i}$</td>
</tr>
</tbody>
</table>
## Variants of TF-IDF

### Recommended tf-idf weighting schemes

<table>
<thead>
<tr>
<th>Weighting Scheme</th>
<th>Document Term Weight</th>
<th>Query Term Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$f_{i,j} \times \log \frac{N}{n_i}$</td>
<td>$(0.5 + 0.5 \frac{f_{i,q}}{\max_i f_{i,q}}) \times \log \frac{N}{n_i}$</td>
</tr>
<tr>
<td>2</td>
<td>$1 + \log f_{i,j}$</td>
<td>$\log(1 + \frac{N}{n_i})$</td>
</tr>
<tr>
<td>3</td>
<td>$(1 + \log f_{i,j}) \times \log \frac{N}{n_i}$</td>
<td>$(1 + \log f_{i,q}) \times \log \frac{N}{n_i}$</td>
</tr>
</tbody>
</table>
TF-IDF Properties

- Tf, idf, and tf-idf weights for the *Wall Street Journal* reference collection sorted by decreasing tf weights

- To represent the $t_f$ weight in the graph, we sum the term frequencies across all documents
  - That is, we used the term collection frequency $F_i$

- Plotted in logarithmic scale
TF-IDF Properties

Weights used on graph:

\[ tf_i = 1 + \log \sum_{j=1}^{N} f_{i,j} \quad idf_i = \log \frac{N}{n_i} \]

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.
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**
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

Documents represented as vectors of weighted terms

- Each term of a collection is associated with an orthonormal unit vector $\vec{k}_i$ in a t-dimensional space.
- For each term $k_i$ of a document $d_j$ is associated the term vector component $w_{i,j} \times \vec{k}_i$.

Retrieval Evaluation, Modern Information Retrieval, Addison Wesley, 2006 – p. 52
Document Length Normalization

The document representation $\vec{d}_j$ 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

$$|\vec{d}_j| = \sqrt{\sum_{i}^{t} w_{i,j}^2}$$
Document Length Normalization

Three variants of document lengths for the example collection

<table>
<thead>
<tr>
<th>Document</th>
<th>Size in bytes</th>
<th>Number of words</th>
<th>Vector norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>34</td>
<td>10</td>
<td>5.068</td>
</tr>
<tr>
<td>$d_2$</td>
<td>37</td>
<td>11</td>
<td>4.899</td>
</tr>
<tr>
<td>$d_3$</td>
<td>41</td>
<td>10</td>
<td>3.762</td>
</tr>
<tr>
<td>$d_4$</td>
<td>43</td>
<td>12</td>
<td>7.738</td>
</tr>
</tbody>
</table>
Classic IR Models
The Vector Model
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

$$\vec{d}_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$$
$$\vec{d}_q = (w_{1q}, w_{2q}, \ldots, w_{tq})$$
The Vector Model

Similarity

\[ \text{sim}(d_j, q) = \cos(\theta) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}} \]

Since \( w_{ij} > 0 \) and \( w_{iq} > 0 \) then \( 0 \leq \text{sim}(d_j, q) \leq 1 \)

A document is retrieved even if it matches the query terms only partially
Weights in the Vector model are basically tf-idf weights

\[
w_{i,q} = (1 + \log f_{i,q}) \times \log \frac{N}{n_i}
\]

\[
w_{i,j} = (1 + \log f_{i,j}) \times \log \frac{N}{n_i}
\]

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”

<table>
<thead>
<tr>
<th>doc</th>
<th>rank computation</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>$1 \times 3 + 0.415 \times 0.830$</td>
<td>0.660</td>
</tr>
<tr>
<td>$d_2$</td>
<td>$1 \times 2 + 0.415 \times 0$</td>
<td>0.408</td>
</tr>
<tr>
<td>$d_3$</td>
<td>$1 \times 0 + 0.415 \times 1.073$</td>
<td>0.118</td>
</tr>
<tr>
<td>$d_4$</td>
<td>$1 \times 0 + 0.415 \times 1.073$</td>
<td>0.058</td>
</tr>
</tbody>
</table>
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 degree of similarity to the query
- document length normalization is naturally built-in into the ranking

Disadvantages:
- It assumes independence of index terms
Classic IR Models
Probabilistic Model
The probabilistic model capture 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 as specification of the properties of this ideal answer set.

But, what are these properties?
Probabilistic Model

- An initial set of documents is retrieved somehow
- User inspects these docs looking for the relevant ones (in truth, only top 10-20 need to be inspected)
- IR system uses this information to refine description of ideal answer set
- By repeating this process, it is expected that the description of the ideal answer set will improve
Probabilistic Ranking Principle

The probabilistic model

- Tries to estimate the probability that a document will be relevant to a user query.
- Assumes that this probability depends on the query and the document representations only.
- Ideal answer set is referred to as $R$ and should maximize the probability of relevance.

But,

- How to compute these probabilities?
- What is the sample space?
The Ranking

Let,

- $R$ be the set of relevant documents to the query $q$
- $\overline{R}$ be the set of non-relevant documents
- $P(R|\vec{d}_j)$ 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

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

Using Bayes’ rule,

\[
sim(d_j, q) = \frac{P(\vec{d}_j | R, q) \times P(R, q)}{P(\vec{d}_j | \bar{R}, q) \times P(\bar{R}, q)} \sim \frac{P(\vec{d}_j | R, q)}{P(\vec{d}_j | \bar{R}, q)}
\]

where

- \( P(\vec{d}_j | R, q) \): probability of randomly selecting the document \( d_j \) from the set \( R \)
- \( P(R, q) \): probability that a document randomly selected from the entire collection is relevant to query \( q \)
- \( P(\vec{d}_j | \bar{R}, q) \) and \( P(\bar{R}, q) \): analogous and complementary
The Ranking

Assuming that the weights $w_{i,j}$ are binary values and assuming the independence among the index terms:

$$\text{sim}(d_j, q) \sim \frac{\left(\prod_{k_i|w_{i,j}=1} P(k_i|R, q)\right) \times \left(\prod_{k_i|w_{i,j}=0} P(k_i|R, q)\right)}{\left(\prod_{k_i|w_{i,j}=1} P(k_i|R, q)\right) \times \left(\prod_{k_i|w_{i,j}=0} P(k_i|R, q)\right)}$$

where

- $P(k_i|R, q)$: probability that the term $k_i$ is present in a document randomly selected from the set $R$
- $P(\bar{k_i}|R, q)$: probability that $k_i$ is not present in a document randomly selected from the set $R$
- probabilities with $\bar{R}$: analogous to the ones just described
To simplify our notation, let us adopt the following conventions

\[ p_{iR} = P(k_i | R, q) \]
\[ q_{iR} = P(k_i | \overline{R}, q) \]

Since

\[ P(k_i | R, q) + P(k_i | \overline{R}, q) = 1 \]
\[ P(k_i | \overline{R}, q) + P(k_i | \overline{R}, q) = 1 \]

then we can write:

\[
\text{sim}(d_j, q) \sim \frac{\left( \prod_{k_i | w_{i,j}=1} p_{iR} \right) \times \left( \prod_{k_i | w_{i,j}=0} (1 - p_{iR}) \right)}{\left( \prod_{k_i | w_{i,j}=1} q_{iR} \right) \times \left( \prod_{k_i | w_{i,j}=0} (1 - q_{iR}) \right)}
\]
The Ranking

Taking logarithms, we write

\[ \text{sim}(d_j, q) \sim \log \prod_{k_i | w_{i,j}=1} p_{iR} + \log \prod_{k_i | w_{i,j}=0} (1 - p_{iR}) \]

\[ - \log \prod_{k_i | w_{i,j}=1} q_{iR} - \log \prod_{k_i | w_{i,j}=0} (1 - q_{iR}) \]
The Ranking

Summing up terms that cancel each other, we obtain

\[ \text{sim}(d_j, q) \sim \log \prod_{k_i | w_{i,j} = 1} p_{iR} + \log \prod_{k_i | w_{i,j} = 0} (1 - p_{ir}) \]

\[ - \log \prod_{k_i | w_{i,j} = 1} (1 - p_{ir}) + \log \prod_{k_i | w_{i,j} = 1} (1 - p_{ir}) \]

\[ - \log \prod_{k_i | w_{i,j} = 1} q_{iR} - \log \prod_{k_i | w_{i,j} = 0} (1 - q_{iR}) \]

\[ + \log \prod_{k_i | w_{i,j} = 1} (1 - q_{iR}) - \log \prod_{k_i | w_{i,j} = 1} (1 - q_{iR}) \]
The Ranking

Using logarithm operations, we obtain

\[ \text{sim}(d_j, q) \sim \log \prod_{k_i | w_{i,j}=1} \frac{p_iR}{(1 - p_iR)} + \log \prod_{k_i} (1 - p_iR) \]

\[ + \log \prod_{k_i | w_{i,j}=1} \frac{(1 - q_iR)}{q_iR} - \log \prod_{k_i} (1 - q_iR) \]

Notice that two terms are constants for all index terms, and can be disregarded for the purpose of ranking
Assuming that
\[ \forall k_i \not\in q, \ p_iR = q_iR \]
and converting the log products into sums of logs, we finally obtain

\[
sim(d_j, q) \sim \sum_{k_i \in q \land k_i \in d_j} \log \left( \frac{p_iR}{1 - p_iR} \right) + \log \left( \frac{1 - q_iR}{q_iR} \right)
\]

which is a key expression for ranking computation in the probabilistic model.
Let,

- $N$ be the number of document 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 in these values, we can build the following contingency table

<table>
<thead>
<tr>
<th></th>
<th>relevant</th>
<th>non-relevant</th>
<th>all docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>docs that contain $k_i$</td>
<td>$r_i$</td>
<td>$n_i - r_i$</td>
<td>$n_i$</td>
</tr>
<tr>
<td>docs that do not contain $k_i$</td>
<td>$R - r_i$</td>
<td>$N - n_i - (R - r_i)$</td>
<td>$N - n_i$</td>
</tr>
<tr>
<td>all docs</td>
<td>$R$</td>
<td>$N - R$</td>
<td>$N$</td>
</tr>
</tbody>
</table>
Term Incidence Contingency Table

If the information of the values of the contingency table were available for any given query, we could write

\[ p_{iR} = \frac{r_i}{R} \]

\[ q_{iR} = \frac{n_i - r_i}{N - R} \]

Then, the equation expression for ranking computation in the probabilistic model can be rewritten as

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

where \( k_i[q,d_j] \) is a short notation for \( k_i \in q \land k_i \in d_j \)
For the previous formula, we are still dependent on estimating what are the relevant dos for the query.

For handling small values of $r_i$, we add $0.5$ to each of the terms in the formula above, which yields:

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

This formula is considered as the classic ranking equation for the probabilistic model.

It is known as the Robertson-Sparck Jones equation.
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 bootstrap the ranking equation, which leads to

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

This equation provides an idf-like ranking formula.

In the absence of relevance information, this is the equation for ranking in the probabilistic model.
Term Incidence Contingency Table

Document ranks computed by the previous probabilistic ranking equation for the query “to do”

<table>
<thead>
<tr>
<th>doc</th>
<th>rank computation</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>$\log \frac{4-2+0.5}{2+0.5} + \log \frac{4-3+0.5}{3+0.5}$</td>
<td>-1.222</td>
</tr>
<tr>
<td>$d_2$</td>
<td>$\log \frac{4-2+0.5}{2+0.5}$</td>
<td>0</td>
</tr>
<tr>
<td>$d_3$</td>
<td>$\log \frac{4-3+0.5}{3+0.5}$</td>
<td>-1.222</td>
</tr>
<tr>
<td>$d_4$</td>
<td>$\log \frac{4-3+0.5}{3+0.5}$</td>
<td>-1.222</td>
</tr>
</tbody>
</table>
The previous probabilistic ranking equation produced negative weights by the term “do.”

This 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:

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

In this Equation, a term that occurs in all documents \((n_i = N)\) produces a weight equal to zero.
Using this formulation, we redo the ranking computation for our example collection for the query “to do”.

<table>
<thead>
<tr>
<th>doc</th>
<th>rank computation</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>$\log \frac{4+0.5}{2+0.5} + \log \frac{4+0.5}{3+0.5}$</td>
<td>1.210</td>
</tr>
<tr>
<td>$d_2$</td>
<td>$\log \frac{4+0.5}{2+0.5}$</td>
<td>0.847</td>
</tr>
<tr>
<td>$d_3$</td>
<td>$\log \frac{4+0.5}{3+0.5}$</td>
<td>0.362</td>
</tr>
<tr>
<td>$d_4$</td>
<td>$\log \frac{4+0.5}{3+0.5}$</td>
<td>0.362</td>
</tr>
</tbody>
</table>
Our examples above considered that $r_i = R = 0$

An alternative is to estimate $r_i$ and $R$ performing an initial search:

- select the top 10-20 ranked documents
- inspect them to gather new estimates for $r_i$ and $R$
- remove the 10-20 documents used from the collection
- rerun the query with the estimates obtained for $r_i$ and $R$

Unfortunately, procedures such as these require human intervention to initially select the relevant documents
Improving the Initial Ranking

Consider the equation

\[ \text{sim}(d_j, q) \sim \sum_{k_i \in q \land k_i \in d_j} \log \left( \frac{p_{iR}}{1 - p_{iR}} \right) + \log \left( \frac{1 - q_{iR}}{q_{iR}} \right) \]

How obtain the probabilities \( p_{iR} \) and \( q_{iR} \)?

Estimates based on assumptions:

- \( p_{iR} = 0.5 \)
- \( q_{iR} = \frac{n_i}{N} \) where \( n_i \) is the number of docs that contain \( k_i \)

Use this initial guess to retrieve an initial ranking

Improve upon this initial ranking
Improving the Initial Ranking

Substituting $p_{iR}$ and $q_{iR}$ into the previous Equation, we obtain:

$$sim(d_j, q) \sim \sum_{k_i \in q \land k_i \in d_j} \log \left( \frac{N - n_i}{n_i} \right)$$

That is the equation used when no relevance information is provided, without the 0.5 correction factor.

Given this initial guess, we can provide an initial probabilistic ranking.

After that, we can attempt to improve this initial ranking as follows.
Improving the Initial Ranking

We can attempt to improve this initial ranking as follows.

Let

- \( D \): set of docs initially retrieved
- \( D_i \): subset of docs retrieved that contain \( k_i \)

Reevaluate estimates:

- \( p_{iR} = \frac{D_i}{D} \)
- \( q_{iR} = \frac{n_i - D_i}{N - D} \)

This process can then be repeated recursively.
Improving the Initial Ranking

\[ \text{sim}(d_j, q) \sim \sum_{k_i \in q \land k_i \in d_j} \log \left( \frac{N - n_i}{n_i} \right) \]

To avoid problems with \( D = 1 \) and \( D_i = 0 \):

\[ p_iR = \frac{D_i + 0.5}{D + 1}; \quad q_iR = \frac{n_i - D_i + 0.5}{N - D + 1} \]

Also,

\[ p_iR = \frac{D_i + \frac{n_i}{N}}{D + 1}; \quad q_iR = \frac{n_i - D_i + \frac{n_i}{N}}{N - D + 1} \]
Pluses and Minuses

Advantages:
- Docs ranked in decreasing order of probability of relevance

Disadvantages:
- Need to guess initial estimates for $p_{iR}$
- Method does not take into account $tf$ factors
- The lack of document length normalization
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.