IR Models.

Johan Bollen
Old Dominion University
Department of Computer Science

jbollen@cs.odu.edu
http://www.cs.odu.edu/~jbollen
Structure

1. IR formal characterization
   (a) Mathematical framework
   (b) Document and keyterm representations

2. Models
   (a) Boolean Model
   (b) Vector Space Model
   (c) Probabilistic Model
Formal Characterization of IR systems

An information retrieval model is defined as the quadruple:

\[ [\mathcal{D}, \mathcal{Q}, \mathcal{F}, \mathcal{R}(q_i, d_j)] \]

where

1. \( \mathcal{D} \) is a set of document representations (logical views)
2. \( \mathcal{Q} \) is a set of representations constituting the user’s information needs or query
3. \( \mathcal{F} \) represents the framework for modeling document representations, queries and their relationships
4. \( \mathcal{R}(q_i, d_j) \) is a ranking function which associates a number \( \in \mathbb{R} \) with a query \( q_i \in \mathcal{Q} \) and a document representation \( d_j \in \mathcal{D} \).
Formal Characterization of IR systems
Classic IR concepts

1. Concepts:
   (a) Documents represented by index terms
   (b) Index term weights: specificity for specific document
   (c) Same may apply for query

2. Relevant to boolean, vector and probabilistic models

   \[ k_i \in \mathcal{K} = \{k_1, \ldots, k_t\} \]: generic index term,
   \( d_j \): document
   \( w_{i,j} > 0 \): weight associated with \((k_i, d_j)\)
   if \( k_i \) not in \( d_j \), \( w_{i,j} = 0 \)
   \( d_j \) is thus associated with index term vector
   \( \vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j}) \)
   function \( g_i \) return weight of index \( k_i \) in \( \vec{d}_j \),
   such that \( g_i(\vec{d}_j) = w_{i,j} \)
Example

Edgar Allen Poe’s The Conqueror Worm, 1843

Last half of last paragraph:

1) And the angels, all pallid and wan,
2) Uprising, unveiling, affirm
3) That the play is the tragedy, ”Man,”
4) And its hero the Conqueror Worm.

Hand selected keyterms:

1) affirm 	 7) play
2) angel 	 8) tragedy
3) conqueror 	 9) unveil
4) hero 	 10) uprise
5) man 	 11) wan
6) pallid 	 12) worm

\[ g_5(\vec{d}_3) = 1, g_3(\vec{d}_2) = 0 \]
Matrix representation

\[
\begin{pmatrix}
  w_{1,1} & w_{2,1} & \cdots & w_{1,t} \\
  w_{2,1} & w_{2,2} & \cdots & w_{2,t} \\
  \cdots \\
  w_{n,1} & w_{n,1} & \cdots & w_{n,t}
\end{pmatrix}
\]

1. Keyterm-document, or in this case document-keyterm, matrix
2. Represents weight values between keyterms and documents
3. Usually sparse, and very large
4. Assumption of independence or orthogonality: keyterms are not related!
5. In this specific case: weight values are binary

\[
d_3 = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)
\]

\[
k_4 = (0, 0, 0, 1)
\]
What are IR models

1. Abstract models of IR techniques:
   (a) characteristics
   (b) Assumptions
   (c) General modus operandi
   (d) Properties

2. Do not comprise full blown IR systems
   (a) Technical details will be discussed later

(b) Focus is assumptions and characteristics

3. Present discussion is not end all be all of IR
   (a) New models can be introduced
   (b) Devil is often in details
   (c) Large collections require specific engineering efforts
Boolean Model

1. Basic principles:
   (a) User query is shaped as Boolean expression
   (b) Documents are predicted to be either relevant or not

2. Advantages:
   (a) Neat formalism
   (b) Easy and efficient implementation
   (c) Widespread adoption has accustomed users to Boolean queries

3. Disadvantages:
   (a) No document ranking
   (b) More like a data retrieval than information retrieval model
   (c) In spite of user preferences, not straightforward to formulate adequate queries
Boolean Model: Formalization

\[ w_{i,j} \in \{0, 1\} \]

Query \( q \) consists of keyterms connected by **NOT**, **AND** and **OR**.

For example, \( q = [k_a \land (k_b \lor \neg k_c)] \)

or: \( q = k_a \&\& (k_b \| \neg k_c) \)

or: “brothers” AND ( “chemical” OR NOT “doobie” )

For every Boolean statement there exists a Disjunctive Normal Form (DNF):

\[ q = [k_a \land (k_b \lor \neg k_c)] = (1, 0, 0) \lor (1, 1, 0) \lor (1, 1, 1) \]

<table>
<thead>
<tr>
<th>( k_a )</th>
<th>( k_b )</th>
<th>( k_c )</th>
<th>( [k_a \land (k_b \lor \neg k_c)] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 ( k_a \land \neg k_b \land \neg k_c )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 ( k_a \land k_b \land \neg k_c )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 ( k_a \land k_b \land k_c )</td>
</tr>
</tbody>
</table>
DNF
Boolean Model: Query matching

Let $\bar{q}_{dnf}$ be DNF for $q$ and $\bar{q}_{cc}$ be the conjunctive components of $\bar{q}_{dnf}$ then

$$sim(d_j, q) = \begin{cases} 1 & \exists \bar{q}_{cc} : (\bar{q}_{cc} \in \bar{q}_{dnf}) \land (\forall k_i, g_i(d_j) = g_i(\bar{q}_{cc})) \\ 0 & \text{otherwise} \end{cases}$$

In human language, the similarity between a document and a user query is one, when there exists a conjunctive component such that every keyterm in the query has the same value in one of its DNF components as it does in the document ($1 =$ present)
Boolean Model: Example

Query: “play” AND (“tragedy” OR NOT “worm”)

translates to DNF:

(“play” AND NOT “tragedy” AND NOT “worm”) = (100)

OR (“play” AND “tragedy” AND NOT “worm”) = (110)

OR (“play” AND “tragedy” AND “warm”) = (111)

1) affirm, 2) angel, 3) conqueror, 4) hero, 5) man, 6) pallid,

7) play, 8) tragedy, 9) unveil, 10) uprise, 11) wan, 12) worm

\[
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

\[d_3 = \text{That the play is the tragedy, ”Man,”}\]
Boolean Model: Discussion

1. Very popular because of its simplicity and efficiency
2. Does not rank results
3. Strongly dependent on exact use of keyterms
4. Not really IR, but rather data retrieval
   (a) keyterms are semantically “shallow” handles on files
   (b) User queries are intended to match completely:
      i. problems with jargon
      ii. synonymy
      iii. user has to be aware of exact keyterms
      iv. humans are not very good at formulating complicated Boolean queries
Vector Space Model

1. Keyterm based model
2. Allows partial matching and ranking of documents
3. Basic principles:
   (a) Document represented by keyterm vector
   (b) $t$ dimensional space spanned by set of keyterms
   (c) Query represented by keyterm vector
   (d) Similarity document-keyterm calculated based on vector distance

4. Requires:
   (a) keyterm weighing for document vectors
   (b) keyterm weighing for query
   (c) Distance measure for document-keyterm vectors

5. Features:
   (a) Efficient
   (b) Successful
   (c) Easy to grasp visual representation
   (d) Applicable to document-document matching
Vector Space Model

**Definition:**

Let

\[ w_{i,j} \text{ be associated with } (k_i, d_j), w_{i,j} \in \mathbb{R}^+ \]

\[ w_{i,q} \text{ be associated with } (k_i, q), w_{i,j} \in \mathbb{R}^+ \]

Query vector \( \vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q}) \)

\( t \) is total number of index terms in system

Document \( d \) is represented by: \( \vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j}) \).

Both document \( d_j \) and query \( q \) are thus represented as \( t \)-dimensional vectors.

The similarity between document and query is evaluated by the cosine of their angle e.g.:

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

Based on definition of dot-product:

\[
d_j \cdot q = \|d_j\| \times \|q\| \cos(\alpha)
\]

 cosine similarity \( \in [0, 1] \) since \( w_{i,j} \geq 0 \) and \( w_{i,j} \geq 0 \)
Cosine similarity measure

\[ \cos(\alpha) = 0: \text{perpendicular vectors} \quad \cos(\alpha) = 1: \text{vectors co-linear} \]
Mommy, where do the index term weights come from?

1. Goal is to find weight that expresses how specific a term is to the intended set of relevant documents

2. Use of frequency

3. Two issues:
   (a) term frequency (TF): frequency within document
   (b) document frequency (DF): frequency within collection

4. Term frequency

      (a) Expresses connection between term and specific document
      (b) Term often occurs within document indicative of its meaning

5. Document Frequency

      (a) Expresses general frequency of term, e.g. “the”
      (b) Characteristic of language

6. Aim is to find keyterms which balance high TF and relatively low DF
Definition

Term frequency

Let $N$ be total number of documents in collection and $n_i$ number of documents in which index term $k_i$ appears.

freq$_{i,j}$ is frequency of $k_i$ in $d_j$.

Normalized term frequency is then defined as:

$$f_{i,j} = \frac{\text{freq}_{i,j}}{\max_l \text{freq}_{l,j}}$$

where $\max_l \text{freq}_{l,j}$ is determined over all terms in document $d_j$.

Inverse document frequency

$$\text{idf}_i = \log \frac{N}{n_i}$$

Index term weight

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$$
**Query keyterm weights**

Balance DF and query frequency:

\[ w_{i,q} = \left( 0.5 + \frac{0.5 \text{freq}_{i,q}}{\max_i \text{freq}_{i,q}} \right) \times \log \frac{N}{n_i} \]

so that \( \text{freq}_{i,q} \) is frequency of term in query (usually 1)

**Document-keyterm matrix:**

<table>
<thead>
<tr>
<th></th>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>( k_3 )</th>
<th>( k_4 )</th>
<th>( k_5 )</th>
<th>( k_6 )</th>
<th>( k_7 )</th>
<th>( k_8 )</th>
<th>( k_9 )</th>
<th>( k_{10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_1 )</td>
<td>( w_{1,1} )</td>
<td>( w_{1,2} )</td>
<td>( \cdots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( w_{1,10} )</td>
</tr>
<tr>
<td>( d_2 )</td>
<td>( w_{2,1} )</td>
<td>( w_{2,2} )</td>
<td>( \cdots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( w_{2,10} )</td>
</tr>
<tr>
<td>( d_3 )</td>
<td>( w_{3,1} )</td>
<td>( w_{3,2} )</td>
<td>( \cdots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( w_{3,10} )</td>
</tr>
<tr>
<td>( d_4 )</td>
<td>( w_{4,1} )</td>
<td>( w_{4,2} )</td>
<td>( \cdots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( w_{4,10} )</td>
</tr>
</tbody>
</table>
Example
Edgar Allen Poe’s The Raven (1845)
First paragraph:

1) Once upon a **midnight** dreary, while I pondered, weak and weary,
2) Over many a quaint and curious **volume** of forgotten **lore**,
3) While I nodded, nearly napping, suddenly there came a **tapping**,
4) As of some one gently rapping, rapping at my **chamber door**.
5) “’Tis some **visiter,**” I muttered, ”tapping at my **chamber door** -
6) Only this, and **nothing** more.”

**Hand selected keyterms - document frequency:**
1) chamber-2 , 2) door-2, 3) lore-1, 4) midnight-1,
5) nothing-1, 6) tap-1, 7) visiter-1, 8) volume-1

**document-keyterm matrix:**

6 × 8 matrix whose entries $w_{i,j} \in \mathbb{R}^+$
Example: Document-Keyterm Matrix

<table>
<thead>
<tr>
<th></th>
<th>chamber</th>
<th>door</th>
<th>lore</th>
<th>midnight</th>
<th>nothing</th>
<th>tap</th>
<th>visiter</th>
<th>volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>$w_{1,1}$</td>
<td>$w_{1,2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$w_{1,8}$</td>
</tr>
<tr>
<td>$d_2$</td>
<td>$w_{2,1}$</td>
<td>$w_{2,2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$w_{2,8}$</td>
</tr>
<tr>
<td>$d_3$</td>
<td>$w_{3,1}$</td>
<td>$w_{3,2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$w_{3,8}$</td>
</tr>
<tr>
<td>$d_4$</td>
<td>$w_{4,1}$</td>
<td>$w_{4,2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$w_{4,8}$</td>
</tr>
<tr>
<td>$d_5$</td>
<td>$w_{4,1}$</td>
<td>$w_{4,2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$w_{5,8}$</td>
</tr>
<tr>
<td>$d_6$</td>
<td>$w_{4,1}$</td>
<td>$w_{4,2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$w_{6,8}$</td>
</tr>
</tbody>
</table>

$k_1$ = “chamber”.

Frequency is zero except for $d_4$ and $d_5$.

$$w_{1,4} = f_{i,j} \times \log \frac{N}{n_i}$$ or

$$w_{1,4} = \frac{\text{freq}_{i,j}}{\max_j \text{freq}_{i,j}} \times \log \frac{N}{n_i}$$

$N = 6$, $n_1 = 2$, $\text{freq}_{i,j} = 1$, $\max_j \text{freq}_{i,j} = 1$,

$$w_{1,4} = \frac{1}{1} \times \log_{10} \frac{6}{2} = 0.47$$
### Example: Document-Keyterm Matrix

**Result:**

<table>
<thead>
<tr>
<th></th>
<th>chamber</th>
<th>door</th>
<th>lore</th>
<th>midnight</th>
<th>nothing</th>
<th>tap</th>
<th>visiter</th>
<th>volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.78</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$d_2$</td>
<td>0</td>
<td>0</td>
<td>0.78</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.78</td>
</tr>
<tr>
<td>$d_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.78</td>
<td>0</td>
</tr>
<tr>
<td>$d_4$</td>
<td>0.48</td>
<td>0.48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$d_5$</td>
<td>0.48</td>
<td>0.48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.78</td>
<td>0</td>
</tr>
<tr>
<td>$d_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.78</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Note:**

Weight values for chamber and door are lower because less specific to document.
Example: Query

Query:
“Visitor at your door”

Translation to known keyterms:
{ “visiter”, “door” }

Query weights:
\[ w_{i,q} = \left( 0.5 + \frac{0.5 \text{freq}_{i,q}}{\max_{l} \text{freq}_{l,q}} \right) \times \log \frac{N}{n_i} \]

“visitor” \[ w_{1,q} = \left( 0.5 + \frac{0.5 \times 1}{1} \right) \times \log \frac{6}{1} = 0.195 \]

“door” \[ w_{1,q} = \left( 0.5 + \frac{0.5 \times 1}{1} \right) \times \log \frac{6}{2} = 0.119 \]

Resulting query vector \( (0, 0.119, 0, 0, 0, 0, 0.195, 0) \)
Retrieval

Query - Document similarity:
\[ \text{sim}(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{\|\vec{d}_j\| \times \|\vec{q}\|} \]

Procedure:

1. Calculate Numerator (inner product of \( q \) and document vector) for all documents
   (a) Weight matrix - query vector multiplication = vector of inner products
   (b) Store resulting vector

2. Calculate Denominator (norm of \( q \) and document vectors)

3. Determine similarities
   (a) Normalize vector of inner products by determined vector norms
   (b) Rank documents according to values in resulting vector
Example

Inner Products:

\[
\begin{pmatrix}
0 & 0 & 0 & 0.78 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
0.78 \\
0 \\
0 \\
0 \\
0.78 \\
0 \\
0
\end{pmatrix}
\times
\begin{pmatrix}
0 \\
0.12 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
0 \\
0 \\
0.06 \\
0.21 \\
0
\end{pmatrix}
\]
Example

Norm of query vector:

\[ \sqrt{|\vec{q}| \cdot \vec{q}'} = 0.233 \]

Norms of document vectors:

\[ |\vec{d}_1| = 0.608, \text{ etc.} \]

\((0.78, 1.103, 0.78, 0.679, 1.034, 0.78)\)

Inner product document-query vectors:

\[
\begin{pmatrix}
0 \\
0 \\
0 \\
0.06 \\
0.21 \\
0
\end{pmatrix}
\rightarrow \text{normalize} \rightarrow
\begin{pmatrix}
0 \\
0 \\
0 \\
0.379 \\
0.871 \\
0
\end{pmatrix}
\rightarrow \text{rank} \rightarrow
\]

\[
\begin{array}{c|c}
\text{rank} & \text{doc} \\
1 & d_5 \\
2 & d_6 \\
\ldots
\end{array}
\]
Document to document similarities

Vector space model can be used to calculate similarities between documents in collection. Instead of query to document vector similarity, document to document.

**Procedure:**

1. Rather than operating on query to keyterm, document-document vector
2. Use document-keyterm matrix
   (a) Document vector in our example: row vectors
   (b) Pairwise cosine similarity
3. Procedure is very expensive
   (a) Can be optimized for sparse matrices
   (b) Efficient in comparison to other methods

**In mathematical terms:**

Let $M$ be document-keyterm matrix.
Matrix of inner products $M_i = M \times M^t$
Norms of $M$’s row vectors. etc.

We define $\text{sim}(d_i, d_j) = 1$
Document to document example

\[ M_i = M \times M' = \begin{pmatrix} 0.608 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.217 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.608 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.46 & 0.46 & 0 \\ 0 & 0 & 0 & 0.46 & 1.069 & 0 \\ 0 & 0 & 0 & 0 & 0.608 & 0 \end{pmatrix} \]

Norms of matrix row vectors (denominator for calculation of weight values):

\((0.78, 1.103, 0.78, 0.679, 1.034, 0.78)\)

Normalize matrix values:

\[ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.655 & 0 \\ 0 & 0 & 0 & 0.655 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \]

We find document 4 and 5 to be similar
Sparse matrices

1. Document-keyterm and document-document matrices are necessarily sparse
   (a) Low ratio of non-zero entries over all entries
   (b) Expressed as percentage, e.g. 3% density

2. Sparseness is natural
   (a) Possible matrix entries $n^2$
   (b) Few “relationships” can be semantically valid
   (c) Large, heterogeneous document collections
     (d) Many matrices well below 1%

3. Storage and manipulation techniques
   (a) Coordinate system
   (b) Compressed Column and Compressed Row Format
   (c) Harwell-Boeing format
   (d) Formats only store non-zero entries
      i. dense format: $n^2$
      ii. Harwell-Boeing: $\pm n + 1 + 2nz$
Coordinate system

\[
\begin{pmatrix}
0 & 2 & 5 & 0 & 0 \\
1 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 2 \\
0 & 1 & 0 & 0 & 0 \\
3 & 0 & 0 & 2 & 0
\end{pmatrix}
\]

Represented by:

<table>
<thead>
<tr>
<th>row</th>
<th>2</th>
<th>5</th>
<th>1</th>
<th>4</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>col</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>values</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Compressed Column format or Harwell-Boeing Format

1. Slightly more efficient than coordinate system (about 30% savings)

2. Represent a $n \times m$ matrix by three arrays:
   (a) Column pointers: $m + 1$ values
   (b) Row indexes: $nz$ values
   (c) Values (non-zero entries) $nz$ values

3. Values are retrieved:
   (a) Column pointers lookup
   (b) Row index range lookup
   (c) Value lookup
Harwell-Boeing Format

\[
\begin{pmatrix}
0 & 2 & 5 & 0 & 0 \\
1 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 2 \\
0 & 1 & 0 & 0 & 0 \\
3 & 0 & 0 & 2 & 0 \\
\end{pmatrix}
\]

Represented by:

\[
\begin{array}{c|cccccc}
\text{colptr} & 1 & 3 & 5 & 6 & 8 & 9 \\
\hline
\text{rowindex} & 2 & 5 & 1 & 4 & 1 & 2 & 5 & 3 \\
\text{values} & 1 & 3 & 2 & 1 & 5 & 3 & 2 & 2 \\
\end{array}
\]

1. Specific:
   (a) colptr[3] < rowindex < colptr[3+1] = 5
   (b) rowindex[5] = 1,
   (c) corresponding value = 5

2. General
   (a) value[i][j] = val[colptr[j] ≤ rowindex < colptr[j+1], i]
   (b) note: colptr specifies a range of values in rowindex
   (c) colptr: value index of first value to start a new column
Perl sparse matrix example

#! /usr/bin/perl

# indexes start at zero
@colptr = (0, 2, 4, 5, 7, 8);
@rowindex = (1, 4, 0, 3, 0, 1, 4, 2);
@values = (1, 3, 2, 1, 5, 3, 2, 2);

$i = $ARGV[0];
$j = $ARGV[1];

$r = $colptr[$j+1]−$colptr[$j];

$value = 0;
for ($n=$colptr[$j]; $n<$colptr[$j+1]; $n++){  
    if ( $rowindex[$n] == $i){  
        $value = $values[$n];  
    }
}

print "value: \$value\n";
Perl sparse matrix - Compressed Row Storage (CRS)

```perl
#! /usr/bin/perl

# CRS, indexes start at zero

@rowptr = (0, 2, 4, 5, 6, 8);
@colindex = (1, 2, 0, 3, 4, 1, 0, 3);
@values = (2, 5, 1, 3, 2, 1, 3, 2);

$i = $ARGV[0];
$j = $ARGV[1];

$value = 0;
for ($n=$rowptr[$i]; $n<$rowptr[$i+1]; $n++){
    if ( $colindex[$n] == $j){
        $value = $values[$n];
    }
}

print "value: $value
";
```
Perl sparse matrix - vector multiplication

#!/usr/bin/perl

# CRS, indexes start at zero

@vec = (1, 2, 3, 4, 5);
@rowptr = (0, 2, 4, 5, 6, 8);
@colindex = (1, 2, 0, 3, 4, 1, 0, 3);
@values = (2, 5, 1, 3, 2, 1, 3, 2);

$i = $ARGV[0];

$value = 0;

for ($i=0; $i<5; $i++){
    $sum = 0;
    for ($n=$rowptr[$i]; $n<=$rowptr[$i+1]; $n++){
        $sum += $values[$n] * $vec[$colindex[$n]];
    }
    print $sum . "\n";
}
Vector Space Model conclusion

1. Long History: since 1960s

2. Very efficient
   (a) Use of sparse matrix methods
   (b) Simple linear algebra
   (c) Well proven

3. Flexible:
   (a) Use in query resolution
   (b) Document to document similarity
   (c) Clustering (more later)

4. For these reasons, very popular and often used

5. Disadvantages:
   (a) No clearly defined theoretical framework
   (b) Generation of adequate indexes
   (c) Assumption of index term independence
   (d) Scalability?
The use of probability theory in IR

1. IR is all about uncertainty
   (a) Give user query, will results be relevant or not?
   (b) System may make estimate, but certainty can not be obtained
   (c) Probability of relevancy is issue

2. Bayesesian probability theory:
   (a) relates to the collection of evidence and making informed guesses
   (b) evidence: was document containing certain keyterms relevant?
   (c) evidence: was document containing certain keyterms found not relevant?
   (d) Adjust estimate of probability not on the basis of frequency, but on the basis of evidence
Probabilistic Models

1. Attempts to predict page relevance on probabilistic grounds

2. Rather than compare specific page features (index terms)

3. Provide a clear and well-grounded framework for IR
   (a) Based on statistical principles: very appropriate to IR!
   (b) Relevancy odds can be updated to take into account other factors than text content
   (c) Possibility of user feedback to be fed into system

4. Basic idea:
   (a) Query is associated with ideal answer set
   (b) Properties of answer \(\rightarrow\) index terms
   (c) Initial guess at answer set properties \(\rightarrow\) index terms
   (d) Initial retrieval results
   (e) User feedback can finetune relevancy probabilities
Bayes’ rule

Bayesian probability:
Change your beliefs on the basis of evidence.

Example 1:
Assume two events $a$ (rain) and $b$ (clouds).

$$P(a|b) = \frac{P(b|a) \times P(a)}{P(b)}$$

$P(a|b)$ probability it will rain we see clouds
$P(b|a)$ how often have we seen clouds when it rains?
$P(a)$ how often does it rain?
$P(b)$ how often have we seen clouds?

Example 2:

$a \rightarrow $”Person has cancer”
$b \rightarrow $”Person is a smoker”

$P(a|b) = ?$

In Belgium: $P(b) = 0.3.$

Among those diagnosed with cancer, about 80% smokes, so $P(b|a) = 0.8.$

Overall probability of cancer $P(a) = 0.05$

$$P(a|b) = \frac{P(b|a) \times P(a)}{P(b)} = \frac{0.8 \times 0.05}{0.3} = 0.13$$

Quit now!
Some basic notions of applying probability theory to IR

Can we do the same for IR?

$d_j$ is document in collection.

$R$ represents set of relevant documents.

$\bar{R}$ represents set of non-relevant documents.

Then we can say things like:

$P(d_j|R) = \text{probability that if a document is relevant, it is } d_j$

We need to determine $P(R|d_j)$ from:

1. $P(R)$

2. $P(d_j|C)$

Since queries are specified we want to translate these statements, to index term based statements, for example:

$P(k_i|R) = \text{probability that document containing } k_i \text{ is relevant.}$

Great! So how do we determine all these probabilities?
Probabilistic Models: Formalization

Basic probabilistic notions

\( w_{i,j} \in \{0, 1\} \), \( w_{i,q} \in \{0, 1\} \)

\( R \) is the set of all relevant documents.

\( \overline{R} \) is \( R \)'s complement, i.e. the set of all non-relevant documents.

\( P(R|\vec{d}_j) \) is the probability that \( d_j \) is relevant to query \( q \) or probability of relevancy given \( d_j \).

\( P(\overline{R}|\vec{d}_j) \) is the probability that \( d_j \) is not relevant to \( q \).

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

Bayes’ rule: \( P(r|e) = \frac{P(e|r) \times P(R)}{P(e)} \):

\[
\text{sim}(d_{j,q}) = \frac{P(\vec{d}_j|R) \times P(R)}{P(\vec{d}_j|R) \times P(\overline{R})}
\]

\( P(\vec{d}_j|R) \): probability of selection \( d_j \) from set of relevant documents.

\( P(R) \): probability that randomly selected document is relevant. As book mentions: \( P(R) \) and \( P(\overline{R}) \) are constant for a given collection, so:

\[
\text{sim}(d_j, q) \sim \frac{P(\vec{d}_j|R)}{P(\vec{d}_j|R)}
\]
Probabilistic Models: Formalization (Continued)

Moving to keyterm level:

\[
\text{sim}(d_j, q) \sim \frac{\left(\prod_{g_i(d_j)=1} P(k_i|R)\right) \times \left(\prod_{g_i(d_j)=0} P(\bar{k}_i|R)\right)}{\left(\prod_{g_i(d_j)=1} P(k_i|R)\right) \times \left(\prod_{g_i(d_j)=0} P(\bar{k}_i|R)\right)}
\]

where \(P(k_i|R)\) stands for the probability that \(k_i \in d_j\) selected from \(R\) and \(P(\bar{k}_i|R)\) stands for the probability that \(k_i \notin \text{ind}_j\) selected from \(R\).

\(P(k_i|R)\) stands for our probability to make a mistake, etc.

Finally the canonical probabilistic information retrieval equation:

\[
\text{sim}(d_j, q) \sim \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1-P(k_i|R)} + \log \frac{1-P(\bar{k}_i|R)}{P(\bar{k}_i|R)}\right)
\]

Does this work? No! We need an initial estimate of \(P(k_i|R)\) and \(P(k_i|R)\).

Various schemes.

\[
P(k_i|R) = 0.5
\]

\[
P(k_i|R) = \frac{n_i}{N}
\]

Given \(w_{i,q}\) and \(w_{i,q}\) these estimates allow an initial set of documents to be retrieved and ranked.
Updating our chances

A bit of recursive reasoning:

Let $V$ be a subset of documents initially retrieved and ranked (e.g. rank above threshold).

Let $V_i \subset V$ of documents containing document $k_i$.

We can now improve our guesses for $P(k_i|R)$ and $P(k_i|\overline{R})$:

$$P(k_i|R) = \frac{V_i}{V}$$

$$P(k_i|\overline{R}) = \frac{n_i - V_i}{N - V}$$

So, $P(k|R)$ is updated according to how many of the retrieved set of documents contains $k_i$.

Similarly, the documents not retrieved indicate our chances of a document containing $k_i$ not being relevant.

User assistance can be used to update odds, e.g. user can indicate which documents are relevant.
Updating our chances, contd.

Problem when retrieved set is small:

Adjustment factors:

\[ P(k|R) = \frac{V_i + 0.5}{V + 1} \]

\[ P(k_i|R) = \frac{n_i - V_i + 0.5}{N - V + 1} \]

This factor increases the “importance” of small retrieval sets.

Constant 0.5 factor may not always be desireable:

\[ P(k|R) = \frac{V_i + \frac{n_i}{N}}{V + 1} \]

\[ P(k_i|R) = \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1} \]

so that adjustment is is relative to overall number of documents containing \( k_i \).
Conclusion

1. Boolean Model:
   (a) Binary index term weights
   (b) Disjunctive Normal Factors
   (c) No partial matching
   (d) No ranking
   (e) Simple and very efficient

2. Vector space model
   (a) Index term weights:
      i. Term Frequency
      ii. Inverse Document Frequency
   (b) Query and document vector
   (c) Cosine measure of query-document similarity

3. Probabilistic Model:
   (a) Based on probabilistic estimate of relevancy document given certain keyterms
   (b) Allows system to adapt to users and provides strong formal framework for retrieval
   (c) Rather complicated
   (d) Performance:
      i. Thought to be less efficient than vector space model
      ii. Problematic initial fine-tuning phase
      iii. Probability: semantics may be issue
How do they stack up?

1. Boolean:
   (a) Pro:
      i. Efficient
      ii. Easy to implement
      iii. Prefered by users in spite of inability to formulate adequate queries
   (b) Con:
      i. No ranking
      ii. No partial matching
      iii. Entirely dependent on keyterm definition and binary weights

2. Vector Space Model:
   (a) Pro:
      i. Keyterm weighing
   ii. Partial Matching
   iii. Ranking
   iv. Efficient
   v. Easy to understand framework
   vi. Flexibility
   (b) Con:
      i. Little formal framework
      ii. Assumption of keyterm independence

3. Probabilistic Model:
   (a) Compares poorly to Vector Space Model
   (b) Occam’s razor may apply to IR as well
   (c) Case for many novel models