A Probabilistic model of Information Retrieval

Harsh Thakkar

DA-IICT, Gandhinagar

2015-04-27
Overview

1. Inception
2. Probabilistic Approach to IR
3. Data
4. Basic Probability Theory
5. Probability Ranking Principle
6. Extensions to BIM: Okapi
7. Performance measure
8. Comparision of Models
Why Probability in IR?

- Queries are representations of user’s information need
- Relevance is binary
- Retrieval is inherently uncertain, since the needs of users are vague in nature i.e. change with time
- Probability deals with uncertainty
- Provides a good estimate of which documents to choose, hence more reliable
Outline

1. Inception
2. Probabilistic Approach to IR
3. Data
4. Basic Probability Theory
5. Probability Ranking Principle
6. Extensions to BIM: Okapi
7. Performance measure
8. Comparison of Models
Relevance feedback

In relevance feedback, the user marks documents as relevant/nonrelevant. Given some known relevant and nonrelevant documents, we compute weights for non-query terms that indicate how likely they will occur in relevant documents. Develop a probabilistic approach for relevance feedback and also a general probabilistic model for IR.
Relevance feedback

- In relevance feedback, the user marks documents as relevant/nonrelevant
Relevance feedback

- In relevance feedback, the user marks documents as relevant/nonrelevant.
- Given some known relevant and nonrelevant documents, we compute weights for non-query terms that indicate how likely they will occur in relevant documents.
Relevance feedback

- In relevance feedback, the user marks documents as relevant/nonrelevant
- Given some known relevant and nonrelevant documents, we compute weights for non-query terms that indicate how likely they will occur in relevant documents
- Develop a probabilistic approach for relevance feedback and also a general probabilistic model for IR
Probabilistic Approach to Retrieval

Given a user information need (represented as a query) and a collection of documents (transformed into document representations), a system must determine how well the documents satisfy the query. An IR system has an uncertain understanding of the user query, and makes an uncertain guess of whether a document satisfies the query. Probability theory provides a principled foundation for such reasoning under uncertainty. Probabilistic models exploit this foundation to estimate how likely it is that a document is relevant to a query.
Probabilistic Approach to Retrieval

- Given a user information need (represented as a query) and a collection of documents (transformed into document representations), a system must determine how well the documents satisfy the query.
Probabilistic Approach to Retrieval

- Given a user information need (represented as a query) and a collection of documents (transformed into document representations), a system must determine how well the documents satisfy the query.
  - An IR system has an uncertain understanding of the user query, and makes an uncertain guess of whether a document satisfies the query.
Probabilistic Approach to Retrieval

- Given a user information need (represented as a query) and a collection of documents (transformed into document representations), a system must determine how well the documents satisfy the query.
  - An IR system has an *uncertain understanding* of the user query, and makes an *uncertain guess* of whether a document satisfies the query.
- Probability theory provides a principled foundation for such reasoning under uncertainty.
Probabilistic Approach to Retrieval

- Given a user information need (represented as a query) and a collection of documents (transformed into document representations), a system must determine how well the documents satisfy the query.
  - An IR system has an uncertain understanding of the user query, and makes an uncertain guess of whether a document satisfies the query.
- Probability theory provides a principled foundation for such reasoning under uncertainty.
  - Probabilistic models exploit this foundation to estimate how likely it is that a document is relevant to a query.
Probabilistic IR Models at a Glance
Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
  - Probability ranking principle
Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
  - Probability ranking principle
    - Binary Independence Model, BestMatch25 (Okapi)
Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
  - Probability ranking principle
    - Binary Independence Model, BestMatch25 (Okapi)
- Bayesian networks for text retrieval
Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
  - Probability ranking principle
    - Binary Independence Model, BestMatch25 (Okapi)
- Bayesian networks for text retrieval
- Language model approach to IR
Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
  - Probability ranking principle
    - Binary Independence Model, BestMatch25 (Okapi)
- Bayesian networks for text retrieval
- Language model approach to IR
- Probabilistic methods are one of the oldest but also one of the currently hottest topics in IR
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999
- All collections from TREC 1-7 are used
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999
- All collections from TREC 1-7 are used
- Also older results from datasets such as NPL, UKCIS, Cranfield and a newly created TREC (described below) are reproduced.
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999
- All collections from TREC 1-7 are used
- Also older results from datasets such as NPL, UKCIS, Cranfield and a newly created TREC (described below) are reproduced.
  - Cranfield collection: has short initial manual index descriptions based on the whole document
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999.

- All collections from TREC 1-7 are used.

- Also older results from datasets such as NPL, UKCIS, Cranfield and a newly created TREC (described below) are reproduced.
  - **Cranfield collection**: has short initial manual index descriptions based on the whole document.
  - **NLP**: has short initial automatic descriptions from *abstracts*. 
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999
- All collections from TREC 1-7 are used
- Also older results from datasets such as NPL, UKCIS, Cranfield and a newly created TREC (described below) are reproduced.
  - **Cranfield collection**: has short initial manual index descriptions based on the whole document
  - **NLP**: has short initial automatic descriptions from *abstracts*
  - **UKCIS**: has only the title of the documents
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999
- All collections from TREC 1-7 are used
- Also older results from datasets such as NPL, UKCIS, Cranfield and a newly created TREC (described below) are reproduced.
  - **Cranfield collection**: has short initial manual index descriptions based on the whole document
  - **NLP**: has short initial automatic descriptions from abstracts
  - **UKCIS**: has only the title of the documents
  - **TREC**: has automatic initial descriptions in natural text form mostly from full documents but in some cases from abstracts. TREC is the largest of all the collections
Datasets

- The paper provides a common platform to a variety of performance scattered over many other papers from 1992-1999
- All collections from TREC 1-7 are used
- Also older results from datasets such as NPL, UKCIS, Cranfield and a newly created TREC (described below) are reproduced.
  - **Cranfield collection**: has short initial manual index descriptions based on the whole document
  - **NLP**: has short initial automatic descriptions from *abstracts*
  - **UKCIS**: has only the title of the documents
  - **TREC**: has automatic initial descriptions in natural text form mostly from full documents but in some cases from abstracts. TREC is the largest of all the collections
  - **TREC** had a mixture of Long, Medium and Very short requests (L,M,V)
OLD COLLECTIONS: for further details see Sparck Jones and Webster (1980)

Cranfield 'C14001'
1400 documents in aeronautics with manual word indexing
225 requests, simple sentences or phrases
exhaustive relevance assessments

UKCIS 'U27000Pb'
27361 documents in chemistry represented by titles
75 requests, terms from elaborate SDI profiles
relevance assessments on original profile output

NPL 'N11500A'
11429 documents in electronics represented by titles and abstracts
93 requests, simple sentences or phrases
relevance assessments from original study pooled outputs

NEW COLLECTION: for further details see Harman (1993-7)

TREC 'T741000X'
741856 documents in news, computing, official publications and energy
represented by full text (over 2/3) or abstracts;
these documents are the combined TREC Disc1 and Disc2 sets
150 requests, words from structured profiles with sections
title, description, narrative
'L' long requests = title+description+narrative
'M' medium requests = title+description
'V' very short requests = titles only
these requests are TREC topics 51-200
relevance assessments from Trec evaluation pooled outputs

Figure: Dataset descriptions from (Jones et al., 2000).
## Datasets 3

### SUMMARY STATISTICS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No docs</th>
<th>No terms</th>
<th>Av terms/doc</th>
<th>No reqs</th>
<th>Av terms/req</th>
<th>No reldocs</th>
<th>Av reldocs/req</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI4001</td>
<td>1400</td>
<td>2683</td>
<td>29.9</td>
<td>225</td>
<td>7.9</td>
<td>1614</td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>700</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>780</td>
<td>3.5</td>
</tr>
<tr>
<td>U27000P</td>
<td>27361</td>
<td>n/a</td>
<td>n/a</td>
<td>75</td>
<td>18.3</td>
<td>3739</td>
<td>49.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>13681</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1902</td>
<td>25.4</td>
</tr>
<tr>
<td>N11500A</td>
<td>11429</td>
<td>7491</td>
<td>20.0</td>
<td>93</td>
<td>7.2</td>
<td>2083</td>
<td>22.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>5715</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1061</td>
<td>11.4</td>
</tr>
<tr>
<td>T741000X</td>
<td>741856</td>
<td>1290709*</td>
<td>129.9</td>
<td>150</td>
<td>L 32.6</td>
<td>37819</td>
<td>252.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>M 10.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V 4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>370928</td>
<td>839463</td>
<td></td>
<td></td>
<td></td>
<td>18927</td>
<td>126.1</td>
</tr>
</tbody>
</table>

**Figure:** Dataset statistics (Jones et al., 2000).
Outline

1. Inception
2. Probabilistic Approach to IR
3. Data
4. Basic Probability Theory
5. Probability Ranking Principle
6. Extensions to BIM: Okapi
7. Performance measure
8. Comparision of Models
Basic Probability Theory

For events $A$ and $B$:
- **Joint probability** $P(A \cap B)$ of both events occurring
- **Conditional probability** $P(A \mid B)$ of event $A$ occurring given that event $B$ has occurred

The chain rule gives the fundamental relationship between joint and conditional probabilities:

$$P(A \cap B) = P(A \mid B) P(B) = P(B \mid A) P(A)$$

Similarly for the complement of an event $P(A)$:

$$P(A^{c} \cap B) = P(B \mid A^{c}) P(A^{c})$$

The partition rule: if $B$ can be divided into an exhaustive set of disjoint subcases, then $P(B)$ is the sum of the probabilities of the subcases. A special case of this rule gives:

$$P(B) = P(A \cap B) + P(A^{c} \cap B)$$
Basic Probability Theory

- For events $A$ and $B$
Basic Probability Theory

For events $A$ and $B$

- Joint probability $P(A \cap B)$ of both events occurring
Basic Probability Theory

- For events $A$ and $B$
  - Joint probability $P(A \cap B)$ of both events occurring
  - Conditional probability $P(A|B)$ of event $A$ occurring given that event $B$ has occurred
Basic Probability Theory

- For events $A$ and $B$
  - Joint probability $P(A \cap B)$ of both events occurring
  - Conditional probability $P(A|B)$ of event $A$ occurring given that event $B$ has occurred
- **Chain rule** gives fundamental relationship between joint and conditional probabilities:

$$P(AB) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$
Basic Probability Theory

- For events $A$ and $B$
  - Joint probability $P(A \cap B)$ of both events occurring
  - Conditional probability $P(A|B)$ of event $A$ occurring given that event $B$ has occurred

- **Chain rule** gives fundamental relationship between joint and conditional probabilities:
  \[
  P(AB) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)
  \]

- Similarly for the complement of an event $P(\overline{A})$:
  \[
  P(\overline{A}B) = P(B|\overline{A})P(\overline{A})
  \]
Basic Probability Theory

- For events $A$ and $B$
  - Joint probability $P(A \cap B)$ of both events occurring
  - Conditional probability $P(A|B)$ of event $A$ occurring given that event $B$ has occurred
- **Chain rule** gives fundamental relationship between joint and conditional probabilities:
  \[
P(AB) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)
  \]
- Similarly for the complement of an event $P(\bar{A})$:
  \[
P(\bar{A}B) = P(B|\bar{A})P(\bar{A})
  \]
- **Partition rule**: if $B$ can be divided into an exhaustive set of disjoint subcases, then $P(B)$ is the sum of the probabilities of the subcases. A special case of this rule gives:
  \[
P(B) = P(AB) + P(\bar{A}B)
  \]
Basic Probability Theory

Bayes' Rule for inverting conditional probabilities:

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} \]

Can be thought of as a way of updating probabilities:

Start off with prior probability \( P(A) \) (initial estimate of how likely event \( A \) is in the absence of any other information)

Derive a posterior probability \( P(A|B) \) after having seen the evidence \( B \), based on the likelihood of \( B \) occurring in the two cases that \( A \) does or does not hold

Odds of an event provide a kind of multiplier for how probabilities change:

\[ \text{Odds: } O(A) = \frac{P(A)}{1 - P(A)} \]
Bayes’ Rule for inverting conditional probabilities:

\[
P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[ \frac{P(B|A)}{\sum_{X \in \{A, \overline{A}\}} P(B|X)P(X)} \right] P(A)
\]
Basic Probability Theory

**Bayes’ Rule** for inverting conditional probabilities:

\[
P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[ \frac{P(B|A)}{\sum_{X \in \{A, \bar{A}\}} P(B|X)P(X)} \right] P(A)
\]

Can be thought of as a way of updating probabilities:
Bayes’ Rule for inverting conditional probabilities:

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[ \frac{P(B|A)}{\sum_{X \in \{A, \overline{A}\}} P(B|X)P(X)} \right] P(A) \]

Can be thought of as a way of updating probabilities:

- Start off with prior probability \( P(A) \) (initial estimate of how likely event \( A \) is in the absence of any other information)
Basic Probability Theory

Bayes’ Rule for inverting conditional probabilities:

\[
P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[ \frac{P(B|A)}{\sum_{X \in \{A, \overline{A}\}} P(B|X)P(X)} \right] P(A)
\]

Can be thought of as a way of updating probabilities:

- Start off with prior probability \( P(A) \) (initial estimate of how likely event \( A \) is in the absence of any other information)
- Derive a posterior probability \( P(A|B) \) after having seen the evidence \( B \), based on the likelihood of \( B \) occurring in the two cases that \( A \) does or does not hold
Bayes’ Rule for inverting conditional probabilities:

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[ \frac{P(B|A)}{\sum_{X \in \{A, \overline{A}\}} P(B|X)P(X)} \right] P(A) \]

Can be thought of as a way of updating probabilities:

- Start off with prior probability \( P(A) \) (initial estimate of how likely event \( A \) is in the absence of any other information)
- Derive a posterior probability \( P(A|B) \) after having seen the evidence \( B \), based on the likelihood of \( B \) occurring in the two cases that \( A \) does or does not hold

Odds of an event provide a kind of multiplier for how probabilities change:

\[ \text{Odds: } O(A) = \frac{P(A)}{P(\overline{A})} = \frac{P(A)}{1 - P(A)} \]
Basic Model

- The probability of a document (D) being relevant (L) to a user is modeled as:

\[
P(L|D) = \frac{P(D|L)P(L)}{P(D)}
\]
Basic Model

- The probability of a document (D) being relevant (L) to a user is modeled as:

$$P(L|D) = \frac{P(D|L)P(L)}{P(D)}$$

We use the Log-Odds to quantify the change since it can be derived from probability by an order-preserving transformation. Thus, the equation becomes:

$$\log P(L|D) = \log P(D|L) - \log P(L)$$
Basic Model

- The probability of a document (D) being relevant (L) to a user is modeled as:

\[ P(L|D) = \frac{P(D|L)P(L)}{P(D)} \]

- We use the Log-Odds to quantify the change since it can be derived from probability by an order-preserving transformation.
Basic Model

- The probability of a document (D) being relevant (L) to a user is modeled as:

\[ P(L|D) = \frac{P(D|L)P(L)}{P(D)} \]

- We use the Log-Odds to quantify the change since it can be derived from probability by an order-preserving transformation.
Basic Model

- The probability of a document \((D)\) being relevant \((L)\) to a user is modeled as:

\[
P(L|D) = \frac{P(D|L)P(L)}{P(D)}
\]

- We use the **Log-Odds** to quantify the change since it can be derived from probability by an order-preserving transformation.

- Thus, the equation becomes:
Basic Model

- The probability of a document (D) being relevant (L) to a user is modeled as:

\[ P(L|D) = \frac{P(D|L)P(L)}{P(D)} \]

- We use the Log-Odds to quantify the change since it can be derived from probability by an order-preserving transformation.

- Thus, the equation becomes:
Basic Model

- The probability of a document (D) being relevant (L) to a user is modeled as:

\[ P(L|D) = \frac{P(D|L)P(L)}{P(D)} \]

- We use the Log-Odds to quantify the change since it can be derived from probability by an order-preserving transformation.

- Thus, the equation becomes:

\[ \log \frac{P(L|D)}{P(L|\bar{D})} = \log \frac{P(D|L)P(L)}{P(D|\bar{L})P(L)} \]
Basic Model - Matching Score

\[ MS_{-PRIM}(D) = \log \frac{P(D|L)}{P(D|\overline{L})} - \log \frac{P(L)}{P(\overline{L})} \] (2)

Here we define the concept of matching score: \( MS(D) \)
Basic Model - Matching Score

\[
\text{Basic Model} - \text{Matching Score} = \log \frac{P(D|L)}{P(D|\neg L)} + \log \frac{P(L)}{P(\neg L)}
\]  (1)

- Here we define the concept of matching score: MS(D)
- Since the function is primitive, thus named MS-PRIM(D), which is as follows:

\[
\text{MS} - \text{PRIM}(D) = \log \frac{P(L|D)}{P(L|\neg D)} - \log \frac{P(L)}{P(\neg L)}
\]  (2)
The basic model of probabilistic IR has been built on the "Independence Assumption", stated as:
Basic Model - Independent attributes

- The basic model of probabilistic IR has been built on the "Independence Assumption", stated as:
- "Given relevance, the attributes are statistically independent"
Basic Model - Independent attributes

- The basic model of probabilistic IR has been built on the “Independence Assumption”, stated as:
- "Given relevance, the attributes are statistically independent”
- Thus, from the probability of statistically independent events, we have:
  \[ P(D|L) = \prod P(A_i = a_i|L) \]
  \[ P(D|\bar{L}) = \prod P(A_i = a_i|\bar{L}) \]
Basic Model - Independent attributes

- The basic model of probabilistic IR has been built on the "Independence Assumption", stated as:

- "Given relevance, the attributes are statistically independent"

- Thus, from the probability of statistically independent events, we have:

  \[ P(D|L) = \prod P(A_i = a_i|L) \]

  \[ P(D|\bar{L}) = \prod P(A_i = a_i|\bar{L}) \]

- Here, \( A_i \) is the \( i^{th} \) attribute with value equal to \( a_i \) for that specific document
The MS-PRIM(D) derived earlier can now be written as:

$$MS - PRIM(D) = \sum \log \frac{P(A_i = a_i | L)}{P(A_i = a_i | \overline{L})}$$

(3)
The MS-PRIM(D) derived earlier can now be written as:

$$MS - PRIM(D) = \sum \log \frac{P(A_i = a_i | L)}{P(A_i = a_i | \overline{L})}$$

Thus, we could calculate a score for each document, based on the sum of the products of all independent attributes relating to the document (D).
Basic Model - Independent attributes (2)

- The **MS-PRIM(D)** derived earlier can now be written as:

\[
MS - PRIM(D) = \sum \log \frac{P(A_i = a_i | L)}{P(A_i = a_i | \bar{L})}
\]  

Thus, we could calculate a score for each document, based on the sum of the products of all independent attributes relating to the document (D).

- This function takes into account both the relevant and non-relevant attributes.
Basic Model - Independent attributes (2)

- The **MS-PRIM(D)** derived earlier can now be written as:

\[
MS - PRIM(D) = \sum \log \frac{P(A_i = a_i | L)}{P(A_i = a_i | \overline{L})}
\]  

(3)

- Thus, we could calculate a score for each document, based on the sum of the products of all independent attributes relating to the document (D).

- This function takes into account both the relevant and non-relevant attributes

- For simplicity we take into account the relevant attributes and take others as "zero"
Basic Model - Independent attributes (2)

- The **MS-PRIM(D)** derived earlier can now be written as:

\[
MS - PRIM(D) = \sum \log \frac{P(A_i = a_i | L)}{P(A_i = a_i | \bar{L})}
\]  

(3)

- Thus, we could calculate a score for each document, based on the sum of the products of all independent attributes relating to the document (D).
- This function takes into account both the relevant and non-relevant attributes.
- For simplicity we take into account the relevant attributes and take others as "zero".
- We define a new measure called **MS-BASIC(D)** which is as:

\[
MS - BASIC(D) = MS - PRIM(D) - \sum \log \frac{P(A_i = 0 | L)}{P(A_i = 0 | \bar{L})}
\]
Basic Model - Independent attributes (3)

Plugging in the value of MS-PRIM(D) from equation 4, we have:

\[
MS - BASIC(D) = \sum \log \frac{P(A_i = a_i|L)}{P(A_i = a_i|\bar{L})} - \log \frac{P(A_i = 0|L)}{P(A_i = 0|\bar{L})}
\]  

(4)
Basic Model - Independent attributes (3)

- Plugging in the value of MS-PRIM(D) from equation 4, we have:

\[
MS - BASIC(D) = \sum \log \frac{P(A_i = a_i | L)}{P(A_i = a_i | \overline{L})} - \log \frac{P(A_i = 0 | L)}{P(A_i = 0 | \overline{L})} \tag{4}
\]

- which can be further simplified to:

\[
= \sum \log \frac{P(A_i = a_i | L)P(A_i = 0 | L)}{P(A_i = a_i | \overline{L})P(A_i = 0 | \overline{L})} \tag{5}
\]
Basic Model - Independent attributes (3)

- We define this as our Weight function, \( W(A_i = a_i) \), Thus,

\[
MS - BASIC(D) = \sum W(A_i = a_i)
\]
We define this as our Weight function, $W(A_i = a_i)$. Thus,

$$MS - BASIC(D) = \sum W(A_i = a_i)$$

This function ($W$), provides a weight for each value of each attribute and the matching score for a document is simply the sum of the weights.
Basic Model - Independent attributes (3)

- We define this as our Weight function, $W(A_i = a_i)$, Thus,

$$MS - BASIC(D) = \sum W(A_i = a_i)$$

- This function ($W$), provides a weight for each value of each attribute and the matching score for a document is simply the sum of the weights.

- $W(A_i = 0)$ is always zero, *i.e.* for a randomly chosen term, which is irrelevant to the query, we can reasonably assume the weight to be zero.
Basic Model-Term presence and absence

- We can further simplify the above model by using the case where attribute $A_i$ is simply the presence or absence of a term $t_i$
Basic Model-Term presence and absence

- We can further simplify the above model by using the case where attribute $A_i$ is simply the presence or absence of a term $t_i$
- We denote $P(t_i|present|L)$ by $p_i$ and $P(t_i|present|\bar{L})$ by $\bar{p}_i$
Basic Model-Term presence and absence

- We can further simplify the above model by using the case where attribute $A_i$ is simply the presence or absence of a term $t_i$
- We denote $P(t_i|\text{present}|L)$ by $p_i$ and $P(t_i|\text{present}|\overline{L})$ by $\overline{p_i}$
- Substituting in previous equation, the new weight formula becomes:

$$w_i = \sum \log \frac{p_i(1 - \overline{p_i})}{\overline{p_i}(1 - p_i)}$$

(6)
Basic Model-Term presence and absence

- We can further simplify the above model by using the case where attribute $A_i$ is simply the presence or absence of a term $t_i$.
- We denote $P(t_i\text{present}|L)$ by $p_i$ and $P(t_i\text{present}|L)$ by $\overline{p_i}$.
- Substituting in previous equation, the new weight formula becomes:

$$w_i = \sum \log \frac{p_i(1 - \overline{p_i})}{\overline{p_i}(1 - p_i)} \quad (6)$$

- Hence, the matching score for the document is just the sum of the weights of the terms present.
Basic Model-Term presence and absence

- We can further simplify the above model by using the case where attribute $A_i$ is simply the presence or absence of a term $t_i$
- We denote $P(t_i|\text{present}|L)$ by $p_i$ and $P(t_i|\text{present}|\bar{L})$ by $\bar{p}_i$
- Substituting in previous equation, the new weight formula becomes:

$$w_i = \sum \log \frac{p_i(1-\bar{p}_i)}{\bar{p}_i(1-p_i)}$$ (6)

- Hence, the matching score for the document is just the sum of the weights of the terms present.
- This formula is later used in the BIM termed as $RSV$ function.
Outline

1. Inception
2. Probabilistic Approach to IR
3. Data
4. Basic Probability Theory
5. Probability Ranking Principle
6. Extensions to BIM: Okapi
7. Performance measure
8. Comparison of Models
The Document Ranking Problem
The Document Ranking Problem

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned

Assume binary notion of relevance: $R_d, q$ is a random dichotomous variable, such that $R_d, q = 1$ if document $d$ is relevant w.r.t query $q$, $R_d, q = 0$ otherwise.

Probabilistic ranking orders documents decreasingly by their estimated probability of relevance w.r.t. query: $P(R = 1 | d, q)$.
The Document Ranking Problem

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned.
- Assume binary notion of relevance: $R_{d,q}$ is a random dichotomous variable, such that
The Document Ranking Problem

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned.
- Assume binary notion of relevance: \( R_{d,q} \) is a random dichotomous variable, such that
  - \( R_{d,q} = 1 \) if document \( d \) is relevant w.r.t query \( q \)
The Document Ranking Problem

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned
- Assume binary notion of relevance: \( R_{d,q} \) is a random dichotomous variable, such that
  - \( R_{d,q} = 1 \) if document \( d \) is relevant w.r.t query \( q \)
  - \( R_{d,q} = 0 \) otherwise
The Document Ranking Problem

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned.
- Assume binary notion of relevance: $R_{d,q}$ is a random dichotomous variable, such that
  - $R_{d,q} = 1$ if document $d$ is relevant w.r.t query $q$
  - $R_{d,q} = 0$ otherwise
- Probabilistic ranking orders documents decreasingly by their estimated probability of relevance w.r.t. query: $P(R = 1|d, q)$
The Document Ranking Problem

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned.
- Assume binary notion of relevance: $R_{d,q}$ is a random dichotomous variable, such that
  - $R_{d,q} = 1$ if document $d$ is relevant w.r.t query $q$
  - $R_{d,q} = 0$ otherwise
- Probabilistic ranking orders documents decreasingly by their estimated probability of relevance w.r.t. query: $P(R = 1|d, q)$
- Assume that the relevance of each document is independent of the relevance of other documents.
Probability Ranking Principle (PRP)

**Probability Ranking Principle (PRP)**

If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable.

**PRP in full**

If the IR system’s response to each query is a ranking of the documents in order of decreasing probability of relevance to the query, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.
Probability Ranking Principle (PRP)

- PRP in brief
Probability Ranking Principle (PRP)

- PRP in brief
  - If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable
Probability Ranking Principle (PRP)

- **PRP in brief**
  - If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable.

- **PRP in full**
Probability Ranking Principle (PRP)

- **PRP in brief**
  - If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable.

- **PRP in full**
  - If [the IR] system’s response to each [query] is a ranking of the documents [...] in order of decreasing probability of relevance to the [query], where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.
Binary Independence Model (BIM)
Binary Independence Model (BIM)

- Traditionally used with the PRP

Assumptions:
Binary Independence Model (BIM)

- Traditionally used with the PRP

Assumptions:
- ‘Binary’ (equivalent to Boolean): documents and queries represented as binary term incidence vectors
Binary Independence Model (BIM)

- Traditionally used with the PRP

Assumptions:
- ‘Binary’ (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - E.g., document $d$ represented by vector $\vec{x} = (x_1, \ldots, x_M)$, where $x_t = 1$ if term $t$ occurs in $d$ and $x_t = 0$ otherwise
Binary Independence Model (BIM)

- Traditionally used with the PRP

Assumptions:
- ‘Binary’ (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - E.g., document $d$ represented by vector $\vec{x} = (x_1, \ldots, x_M)$, where $x_t = 1$ if term $t$ occurs in $d$ and $x_t = 0$ otherwise
  - Different documents may have the same vector representation
Binary Independence Model (BIM)

- Traditionally used with the PRP

Assumptions:
- ‘Binary’ (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - E.g., document \(d\) represented by vector \(\vec{x} = (x_1, \ldots, x_M)\), where \(x_t = 1\) if term \(t\) occurs in \(d\) and \(x_t = 0\) otherwise
  - Different documents may have the same vector representation
- ‘Independence’: no association between terms (not true, but practically works - ‘naive’ assumption of Naive Bayes models)
## Binary incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Worscher</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Each document is represented as a **binary vector** $\in \{0, 1\}^{|V|}$. 
Binary Independence Model
Binary Independence Model

To make a probabilistic retrieval strategy precise, need to estimate how terms in documents contribute to relevance

- Find measurable statistics (term frequency, document frequency, document length) that affect judgments about document relevance
Binary Independence Model

To make a probabilistic retrieval strategy precise, need to estimate how terms in documents contribute to relevance

- Find measurable statistics (term frequency, document frequency, document length) that affect judgments about document relevance
- Combine these statistics to estimate the probability $P(R|d,q)$ of document relevance
To make a probabilistic retrieval strategy precise, need to estimate how terms in documents contribute to relevance

- Find measurable statistics (term frequency, document frequency, document length) that affect judgments about document relevance
- Combine these statistics to estimate the probability $P(R|d, q)$ of document relevance
- Next: how exactly we can do this
Binary Independence Model

\[ P(R|d, q) \] is modeled using term incidence vectors as

\[ P(R = 1|\vec{x}, \vec{q}) = P(\vec{x}|R = 1, \vec{q}) P(R = 1|\vec{q}) \]

\[ P(R = 0|\vec{x}, \vec{q}) = P(\vec{x}|R = 0, \vec{q}) P(R = 0|\vec{q}) \]

Use statistics about the document collection to estimate these probabilities.
Binary Independence Model

\[ P(R|d, q) \] is modeled using term incidence vectors as \[ P(R|\vec{x}, \vec{q}) \]

\[ P(R = 1|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})} \]

\[ P(R = 0|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})} \]

- \[ P(\vec{x}|R = 1, \vec{q}) \] and \[ P(\vec{x}|R = 0, \vec{q}) \]: probability that if a relevant or nonrelevant document is retrieved, then that document’s representation is \( \vec{x} \)
Binary Independence Model

\[ P(R|d, q) \] is modeled using term incidence vectors as \( P(R|\vec{x}, \vec{q}) \)

\[
P(R = 1|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})}
\]
\[
P(R = 0|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})}
\]

- \( P(\vec{x}|R = 1, \vec{q}) \) and \( P(\vec{x}|R = 0, \vec{q}) \): probability that if a relevant or nonrelevant document is retrieved, then that document’s representation is \( \vec{x} \)

- Use statistics about the document collection to estimate these probabilities
Binary Independence Model

\( P(R|d, q) \) is modeled using term incidence vectors as \( P(R|\bar{x}, \bar{q}) \)

\[
P(R = 1|\bar{x}, \bar{q}) = \frac{P(\bar{x}|R = 1, \bar{q})P(R = 1|\bar{q})}{P(\bar{x}|\bar{q})}
\]

\[
P(R = 0|\bar{x}, \bar{q}) = \frac{P(\bar{x}|R = 0, \bar{q})P(R = 0|\bar{q})}{P(\bar{x}|\bar{q})}
\]
Binary Independence Model

\( P(R|d, q) \) is modeled using term incidence vectors as \( P(R|\vec{x}, \vec{q}) \)

\[
\begin{align*}
P(R = 1|\vec{x}, \vec{q}) &= \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})} \\
P(R = 0|\vec{x}, \vec{q}) &= \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})}
\end{align*}
\]

- \( P(R = 1|\vec{q}) \) and \( P(R = 0|\vec{q}) \): prior probability of retrieving a relevant or nonrelevant document for a query \( \vec{q} \)
Binary Independence Model

\[ P(R|d, q) \] is modeled using term incidence vectors as \( P(R|x, \bar{q}) \)

\[
\begin{align*}
P(R = 1|x, \bar{q}) &= \frac{P(x|R = 1, \bar{q})P(R = 1|\bar{q})}{P(x|\bar{q})} \\
P(R = 0|x, \bar{q}) &= \frac{P(x|R = 0, \bar{q})P(R = 0|\bar{q})}{P(x|\bar{q})}
\end{align*}
\]

- \( P(R = 1|\bar{q}) \) and \( P(R = 0|\bar{q}) \): prior probability of retrieving a relevant or nonrelevant document for a query \( \bar{q} \)
- Estimate \( P(R = 1|\bar{q}) \) and \( P(R = 0|\bar{q}) \) from percentage of relevant documents in the collection
Binary Independence Model

$P(R|d, q)$ is modeled using term incidence vectors as $P(R|\vec{x}, \vec{q})$

\[
P(R = 1|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})}
\]

\[
P(R = 0|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})}
\]

- $P(R = 1|\vec{q})$ and $P(R = 0|\vec{q})$: prior probability of retrieving a relevant or nonrelevant document for a query $\vec{q}$
- Estimate $P(R = 1|\vec{q})$ and $P(R = 0|\vec{q})$ from percentage of relevant documents in the collection
- Since a document is either relevant or nonrelevant to a query, we must have that:
Binary Independence Model

$P(R|d, q)$ is modeled using term incidence vectors as $P(R|\vec{x}, \vec{q})$

\[
P(R = 1|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})} \quad P(R = 0|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})}\]

- $P(R = 1|\vec{q})$ and $P(R = 0|\vec{q})$: prior probability of retrieving a relevant or nonrelevant document for a query $\vec{q}$
- Estimate $P(R = 1|\vec{q})$ and $P(R = 0|\vec{q})$ from percentage of relevant documents in the collection
- Since a document is either relevant or nonrelevant to a query, we must have that:

\[
P(R = 1|\vec{x}, \vec{q}) + P(R = 0|\vec{x}, \vec{q}) = 1\]
Deriving a Ranking Function for Query Terms (1)

Given a query $q$, ranking documents by $P(R = 1 | d, q)$ is modeled under BIM as ranking them by $P(R = 1 | \vec{x}, \vec{q})$.

Easier: rank documents by their odds of relevance (gives same ranking)

$$O(R | \vec{x}, \vec{q}) = \frac{P(R = 1 | \vec{x}, \vec{q})}{P(R = 0 | \vec{x}, \vec{q})}$$

$$P(R = 0 | \vec{x}, \vec{q}) = P(R = 1 | \vec{q}) P(\vec{x} | R = 1, \vec{q}) P(\vec{x} | \vec{q}) P(R = 0 | \vec{q}) P(\vec{x} | R = 0, \vec{q})$$

$P(R = 1 | \vec{q})$ is a constant for a given query - can be ignored.
Deriving a Ranking Function for Query Terms (1)

- Given a query $q$, ranking documents by $P(R = 1|d, q)$ is modeled under BIM as ranking them by $P(R = 1|\vec{x}, \vec{q})$
- Easier: rank documents by their odds of relevance (gives same ranking)

$$O(R|\vec{x}, \vec{q}) = \frac{P(R = 1|\vec{x}, \vec{q})}{P(R = 0|\vec{x}, \vec{q})} = \frac{P(R=1|\vec{q})P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})} \cdot \frac{P(\vec{x}|R=1, \vec{q})}{P(\vec{x}|R=0, \vec{q})}$$

$$= \frac{P(R = 1|\vec{q})}{P(R = 0|\vec{q})} \cdot \frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})}$$

- $\frac{P(R=1|\vec{q})}{P(R=0|\vec{q})}$ is a constant for a given query - can be ignored
It is at this point that we make the Naive Bayes conditional independence assumption that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

\[
P(\vec{x} | R = 1, \vec{q}) = \prod_{t=1}^{T} P(x_t | R = 1, \vec{q}) P(x_t | R = 0, \vec{q})
\]

So:

\[
O(R | \vec{x}, \vec{q}) = O(R | \vec{q}) \cdot \prod_{t=1}^{T} P(x_t | R = 1, \vec{q}) P(x_t | R = 0, \vec{q})
\]
Deriving a Ranking Function for Query Terms (2)

It is at this point that we make the **Naive Bayes conditional independence assumption** that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

$$
\frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})} = \prod_{t=1}^{M} \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}
$$

So:

$$
O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^{M} \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}
$$
Deriving a Ranking Function for Query Terms (3)

Since each $x_t$ is either 0 or 1, we can separate the terms:

$$O(R | \vec{x}, \vec{q}) = O(R | \vec{q}) \cdot \prod_{t: x_t = 1} P(x_t = 1 | R = 1, \vec{q}) \cdot \prod_{t: x_t = 0} P(x_t = 0 | R = 1, \vec{q})$$

$$\cdot \prod_{t: x_t = 0} P(x_t = 0 | R = 0, \vec{q})$$
Deriving a Ranking Function for Query Terms (3)

Since each $x_t$ is either 0 or 1, we can separate the terms:
Since each $x_t$ is either 0 or 1, we can separate the terms:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t = 1|R = 1, \vec{q})}{P(x_t = 1|R = 0, \vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t = 0|R = 1, \vec{q})}{P(x_t = 0|R = 0, \vec{q})}$$
Let $p_t = P(x_t = 1 | R = 1, \vec{q})$ be the probability of a term appearing in a relevant document.

Let $p_t = P(x_t = 1 | R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document.

Can be displayed as contingency table:

<table>
<thead>
<tr>
<th>Document</th>
<th>Relevant ($R = 1$)</th>
<th>Nonrelevant ($R = 0$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term present</td>
<td>$p_t$</td>
<td>$1 - p_t$</td>
</tr>
<tr>
<td>Term absent</td>
<td>$1$</td>
<td>$1 - p_t$</td>
</tr>
</tbody>
</table>
Deriving a Ranking Function for Query Terms (4)

Let \( p_t = P(x_t = 1|R = 1, \vec{q}) \) be the probability of a term appearing in relevant document.

\[
\begin{array}{c|c|c}
\text{document} & \text{relevant (} R = 1 \text{)} & \text{nonrelevant (} R = 0 \text{)} \\
\hline
\text{Term present} & p_t & p_t \\
\hline
\text{Term absent} & 1 - p_t & 1 - p_t
\end{array}
\]
Let \( p_t = P(x_t = 1|R = 1, \vec{q}) \) be the probability of a term appearing in relevant document.
Let $p_t = P(x_t = 1 | R = 1, \vec{q})$ be the probability of a term appearing in relevant document

Let $\overline{p_t} = P(x_t = 1 | R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document
Deriving a Ranking Function for Query Terms (4)

- Let $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in relevant document
- Let $\bar{p}_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document
Deriving a Ranking Function for Query Terms (4)

- Let $p_t = P(x_t = 1| R = 1, \vec{q})$ be the probability of a term appearing in relevant document
- Let $\overline{p}_t = P(x_t = 1| R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document
- Can be displayed as contingency table:

<table>
<thead>
<tr>
<th>Term present $x_t = 1$</th>
<th>relevant $(R = 1)$</th>
<th>nonrelevant $(R = 0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_t$</td>
<td>$1 - p_t$</td>
<td>$1 - \overline{p}_t$</td>
</tr>
</tbody>
</table>
Deriving a Ranking Function for Query Terms

Additional simplifying assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents.

If \( q_t = 0 \), then \( p_t = p_t \).

Now we need only to consider terms in the products that appear in the query:

\[
O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t: x_t = 1} p_t \cdot \prod_{t: x_t = 0, q_t = 1} 1 - p_t
\]

The left product is over query terms found in the document and the right product is over query terms not found in the document.
Deriving a Ranking Function for Query Terms

Additional simplifying assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents

- If $q_t = 0$, then $p_t = \overline{p_t}$

Now we need only to consider terms in the products that appear in the query:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t: x_t = q_t = 1} \frac{p_t}{\overline{p_t}} \cdot \prod_{t: x_t = 0, q_t = 1} \frac{1 - p_t}{1 - \overline{p_t}}$$
Deriving a Ranking Function for Query Terms

Additional simplifying assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents

- If \( q_t = 0 \), then \( p_t = \overline{p_t} \)

Now we need only to consider terms in the products that appear in the query:

\[
O(R|\overline{x}, \overline{q}) = O(R|\overline{q}) \cdot \prod_{t: x_t=q_t=1} \frac{p_t}{\overline{p_t}} \cdot \prod_{t: x_t=0, q_t=1} \frac{1 - p_t}{1 - \overline{p_t}}
\]

- The left product is over query terms found in the document and the right product is over query terms not found in the document
Deriving a Ranking Function for Query Terms

\[\begin{align*}
O(R | \vec{x}, \vec{q}) &= O(R | \vec{q}) \cdot \prod_{t: x_t = q_t = 1} p_t (1 - p_t) \cdot \prod_{t: q_t = 1} 1 - p_t \\
\text{The left product is still over query terms found in the document, but the right product is now over all query terms, hence constant for a particular query and can be ignored.}
\end{align*}\]

\[\rightarrow \text{The only quantity that needs to be estimated to rank documents w.r.t a query is the left product} \]

\[\text{Hence the Retrieval Status Value (RSV) in this model:}
\]

\[\text{RSV}_d = \log \prod_{t: x_t = q_t = 1} p_t (1 - p_t) = \sum_{t: x_t = q_t = 1} \log p_t (1 - p_t) \cdot p_t (1 - p_t)\]
Deriving a Ranking Function for Query Terms

Including the query terms found in the document into the right product, but simultaneously dividing by them in the left product, gives:

\[ O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t: x_t=q_t=1} \frac{p_t(1-p_t)}{p_t(1-p_t)} \cdot \prod_{t: q_t=1} \frac{1-p_t}{1-p_t} \]

- The left product is still over query terms found in the document, but the right product is now over all query terms, hence constant for a particular query and can be ignored.
Deriving a Ranking Function for Query Terms

Including the query terms found in the document into the right product, but simultaneously dividing by them in the left product, gives:

\[ O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-p_t)}{p_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-p_t} \]

- The left product is still over query terms found in the document, but the right product is now over all query terms, hence constant for a particular query and can be ignored.
- → The only quantity that needs to be estimated to rank documents w.r.t a query is the left product.
Deriving a Ranking Function for Query Terms

Including the query terms found in the document into the right product, but simultaneously dividing by them in the left product, gives:

\[ O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-p_t)}{p_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-p_t} \]

- The left product is still over query terms found in the document, but the right product is now over all query terms, hence constant for a particular query and can be ignored.
- The only quantity that needs to be estimated to rank documents w.r.t a query is the left product
- Hence the Retrieval Status Value (RSV) in this model:

\[ RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-p_t)}{p_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-p_t)}{p_t(1-p_t)} \]
Deriving a Ranking Function for Query Terms

\[ c_t = \log \frac{p_t}{1 - p_t} \]

The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant \( \left( \frac{p_t}{1 - p_t} \right) \), and (ii) the odds of the term appearing if the document is nonrelevant \( \left( \frac{p_t}{1 - p_t} \right) \).

- \( c_t = 0 \): term has equal odds of appearing in relevant and nonrelevant docs.
- \( c_t > 0 \): higher odds to appear in relevant documents.
- \( c_t < 0 \): higher odds to appear in nonrelevant documents.
Deriving a Ranking Function for Query Terms

Equivalent: rank documents using the log odds ratios for the terms in the query \( c_t \):

\[
c_t = \log \frac{p_t(1 - \overline{p_t})}{\overline{p_t}(1 - p_t)} = \log \frac{p_t}{(1 - p_t)} - \log \frac{\overline{p_t}}{1 - \overline{p_t}}
\]

- The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant \( (p_t/(1 - p_t)) \), and (ii) the odds of the term appearing if the document is nonrelevant \( (\overline{p_t}/(1 - \overline{p_t})) \)
Deriving a Ranking Function for Query Terms

Equivalent: rank documents using the log odds ratios for the terms in the query $c_t$:

$$c_t = \log \frac{p_t(1 - \overline{p_t})}{p_t(1 - p_t)} = \log \frac{p_t}{(1 - p_t)} - \log \frac{\overline{p_t}}{1 - p_t}$$

- The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant ($p_t/(1 - p_t)$), and (ii) the odds of the term appearing if the document is nonrelevant ($\overline{p_t}/(1 - \overline{p_t})$)
- $c_t = 0$: term has equal odds of appearing in relevant and nonrelevant docs
Deriving a Ranking Function for Query Terms

Equivalent: rank documents using the log odds ratios for the terms in the query $c_t$:

$$c_t = \log \frac{p_t(1 - \overline{p}_t)}{p_t(1 - p_t)} = \log \frac{p_t}{(1 - p_t)} - \log \frac{\overline{p}_t}{1 - \overline{p}_t}$$

- The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant ($p_t/(1 - p_t)$), and (ii) the odds of the term appearing if the document is nonrelevant ($\overline{p}_t/(1 - \overline{p}_t)$)
- $c_t = 0$: term has equal odds of appearing in relevant and nonrelevant docs
- $c_t$ positive: higher odds to appear in relevant documents
Deriving a Ranking Function for Query Terms

Equivalent: rank documents using the log odds ratios for the terms in the query $c_t$:

$$c_t = \log \frac{p_t (1 - \overline{p_t})}{p_t (1 - p_t)} = \log \frac{p_t}{1 - p_t} - \log \frac{1}{1 - p_t}$$

- The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant ($p_t/(1 - p_t)$), and (ii) the odds of the term appearing if the document is nonrelevant ($\overline{p_t}/(1 - \overline{p_t})$)
- $c_t = 0$: term has equal odds of appearing in relevant and nonrelevant docs
- $c_t$ positive: higher odds to appear in relevant documents
- $c_t$ negative: higher odds to appear in nonrelevant documents
Term weight $c_t$ in BIM
Term weight $c_t$ in BIM

$$c_t = \log \frac{p_t}{1-p_t} - \log \frac{\bar{p_t}}{1-p_t}$$ functions as a term weight.
Term weight $c_t$ in BIM

- $c_t = \log \frac{p_t}{(1-p_t)} - \log \frac{p_t}{1-p_t}$ functions as a term weight.
- Retrieval status value for document $d$: $RSV_d = \sum_{x_t=q_t=1} c_t$. 
Term weight $c_t$ in BIM

- $c_t = \log \frac{p_t}{(1 - p_t)} - \log \frac{\bar{p}_t}{1 - \bar{p}_t}$ functions as a term weight.
- Retrieval status value for document $d$: $RSV_d = \sum_{x_t=q_t=1} c_t$.
- So BIM and vector space model are identical on an operational level . . .
Term weight $c_t$ in BIM

- $c_t = \log \frac{p_t}{(1-p_t)} - \log \frac{p_t}{1-p_t}$ functions as a term weight.
- Retrieval status value for document $d$: $RSV_d = \sum_{x_t=q_t=1} c_t$.
- So BIM and vector space model are identical on an operational level . . .
- . . . except that the term weights are different.
Term weight $c_t$ in BIM

- $c_t = \log \frac{p_t}{(1-p_t)} - \log \frac{p_t}{1-p_t}$ functions as a term weight.
- Retrieval status value for document $d$: $RSV_d = \sum_{x_t=q_t=1} c_t$.
- So BIM and vector space model are identical on an operational level . . .
- . . . except that the term weights are different.
- In particular: we can use the same data structures (inverted index etc) for the two models.
How to compute probability estimates

- For each term $t$ in a query, estimate $c_t$ in the whole collection using a contingency table of counts of documents in the collection, where $n$ is the number of documents that contain term $t$:

<table>
<thead>
<tr>
<th>Document</th>
<th>Relevant</th>
<th>Nonrelevant</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term present</td>
<td>$x_t = 1$</td>
<td>$n - r$</td>
<td>$R - r$</td>
</tr>
<tr>
<td>Term absent</td>
<td>$x_t = 0$</td>
<td>$R - r$</td>
<td>$N - n$</td>
</tr>
</tbody>
</table>

- $p_t = \frac{r}{R}$

- $c_t = K(N,n,R,r) = \log \frac{r}{R - r} \frac{n - r}{(N - n) - (R - r)}$
How to compute probability estimates

For each term $t$ in a query, estimate $c_t$ in the whole collection using a contingency table of counts of documents in the collection, where $n$ is the number of documents that contain term $t$:

<table>
<thead>
<tr>
<th>documents</th>
<th>relevant</th>
<th>nonrelevant</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term present $x_t = 1$</td>
<td>$r$</td>
<td>$n - r$</td>
<td>$n$</td>
</tr>
<tr>
<td>Term absent $x_t = 0$</td>
<td>$R - r$</td>
<td>$N - n - (R - r)$</td>
<td>$N - n$</td>
</tr>
<tr>
<td>Total</td>
<td>$R$</td>
<td>$N - R$</td>
<td>$N$</td>
</tr>
</tbody>
</table>

$$p_t = \frac{r}{R}$$

$$\overline{p_t} = \frac{(n - r)}{(N - R)}$$

$$c_t = K(N, n, R, r) = \log \frac{r/(R - r)}{(n - r)/((N - n) - (R - r))}$$
Avoiding zeros

If any of the counts is a zero, then the term weight is not well-defined. Maximum likelihood estimates do not work for rare events. To avoid zeros: add 0.5 to each count (expected likelihood estimation = ELE). For example, use $R - r + 0.5$ in formula for $R - r$. Thus, formula becomes:

$$SV_d = \sum \log (R - r + 0.5) / (R - r + 0.5) / (n - r + 0.5) / (N - n + 0.5) - (R - r + 0.5)) (7)$$

This was previously known as relevance weight (RW) or matching score relevance weight (MS-RW) (Robertson and Jones, 1976).
Avoiding zeros

- If any of the counts is a zero, then the term weight is not well-defined.
Avoiding zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
Avoiding zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add 0.5 to each count (expected likelihood estimation = ELE)
Avoiding zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add 0.5 to each count (expected likelihood estimation = ELE)
- For example, use $R - r + 0.5$ in formula for $R - r$. Thus, formula becomes:

$$RSV_d = \sum \log \frac{r + 0.5/(R - r + 0.5)}{(n - r + 0.5)/((N - n + 0.5) - (R - r + 0.5))}$$

(7)
Avoiding zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add 0.5 to each count (expected likelihood estimation = ELE)
- For example, use $R - r + 0.5$ in formula for $R - r$. Thus, formula becomes:

\[
RSV_d = \Sigma \log \frac{r + 0.5/(R - r + 0.5)}{(n - r + 0.5)/((N - n + 0.5) - (R - r + 0.5))}
\]

- This was previously known as relevance weight (RW) or matching score relevance weight (MS-RW) (Roberston and Jones, 1976)
Simplifying assumption

Assuming that relevant documents are a very small percentage of the collection, approximate statistics for nonrelevant documents by statistics from the whole collection. Hence, 

\[
(p^t) \text{ (the probability of term occurrence in nonrelevant documents for a query) is } \frac{n}{N}
\]

and

\[
\log\left(\frac{1 - (p^t)}{p^t}\right) \approx \log\left(\frac{N - n}{n}\right)
\]

The above approximation cannot easily be extended to relevant documents.
Simplifying assumption

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for nonrelevant documents by statistics from the whole collection.
Simplifying assumption

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for nonrelevant documents by statistics from the whole collection.
- Hence, $\tilde{(p_t)}$ (the probability of term occurrence in nonrelevant documents for a query) is $n/N$ and

$$\log\left[\frac{1 - \tilde{(p_t)}}{\tilde{p_t}}\right] = \log\left[\frac{N - n}{n}\right] \approx \log \frac{N}{n}$$
Simplifying assumption

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for non-relevant documents by statistics from the whole collection.

- Hence, \( \tilde{p}_t \) (the probability of term occurrence in non-relevant documents for a query) is \( n/N \) and

\[
\log\left(\frac{1 - \tilde{p}_t}{\tilde{p}_t}\right) = \log\left(\frac{N - n}{n}\right) \approx \log N/n
\]

- The above approximation cannot easily be extended to relevant documents.
Probability estimates in relevance feedback
Probability estimates in relevance feedback

- Statistics of relevant documents \( (p_t) \) in relevance feedback can be estimated using maximum likelihood estimation or ELE (add 0.5).
Probability estimates in relevance feedback

- Statistics of relevant documents ($p_t$) in relevance feedback can be estimated using maximum likelihood estimation or ELE (add 0.5).
  - Use the frequency of term occurrence in known relevant documents.
Probability estimates in relevance feedback

- Statistics of relevant documents ($p_t$) in relevance feedback can be estimated using maximum likelihood estimation or ELE (add 0.5).
  - Use the frequency of term occurrence in known relevant documents.
- This is the basis of probabilistic approaches to relevance feedback weighting in a feedback loop.
Probability estimates in relevance feedback

- Statistics of relevant documents \( (p_t) \) in relevance feedback can be estimated using maximum likelihood estimation or ELE (add 0.5).
  - Use the frequency of term occurrence in known relevant documents.
- This is the basis of probabilistic approaches to relevance feedback weighting in a feedback loop.
- What we just saw was a probabilistic relevance feedback exercise since we were assuming the availability of relevance judgments.
Probability estimates in adhoc retrieval

In adhoc retrieval, no user-supplied relevance judgments are available. In this case, $p_t$ is constant over all terms $x$ in the query and $p_t = 0.5$. Each term is equally likely to occur in a relevant document, and so the $p_t$ and $(1 - p_t)$ factors cancel out in the expression for RSV. Weak estimate, but doesn't disagree violently with expectation that query terms appear in many but not all relevant documents.

Combining this method with the earlier approximation for $p_t$, the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting. For short documents (titles or abstracts) in one-pass retrieval situations, this estimate can be quite satisfactory.
Probability estimates in adhoc retrieval

- Ad-hoc retrieval: **no user-supplied relevance judgments available**
Probability estimates in adhoc retrieval

- Ad-hoc retrieval: **no user-supplied relevance judgments available**
- In this case: assume that $p_t$ is constant over all terms $x_t$ in the query and that $p_t = 0.5$
Probability estimates in adhoc retrieval

- Ad-hoc retrieval: **no user-supplied relevance judgments available**
- In this case: assume that $p_t$ is constant over all terms $x_t$ in the query and that $p_t = 0.5$
- Each term is equally likely to occur in a relevant document, and so the $p_t$ and $(1 - p_t)$ factors cancel out in the expression for $RSV$
Probability estimates in adhoc retrieval

- Ad-hoc retrieval: **no user-supplied relevance judgments available**
- In this case: assume that $p_t$ is constant over all terms $x_t$ in the query and that $p_t = 0.5$
- Each term is equally likely to occur in a relevant document, and so the $p_t$ and $(1 - p_t)$ factors cancel out in the expression for $RSV$
- Weak estimate, but doesn’t disagree violently with expectation that query terms appear in many but not all relevant documents
Probability estimates in adhoc retrieval

- Ad-hoc retrieval: no user-supplied relevance judgments available
- In this case: assume that $p_t$ is constant over all terms $x_t$ in the query and that $p_t = 0.5$
- Each term is equally likely to occur in a relevant document, and so the $p_t$ and $(1 - p_t)$ factors cancel out in the expression for $RSV$
- Weak estimate, but doesn’t disagree violently with expectation that query terms appear in many but not all relevant documents
- Combining this method with the earlier approximation for $\bar{p}_t$, the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting
Probability estimates in adhoc retrieval

- Ad-hoc retrieval: **no user-supplied relevance judgments available**
- In this case: assume that $p_t$ is constant over all terms $x_t$ in the query and that $p_t = 0.5$
- Each term is equally likely to occur in a relevant document, and so the $p_t$ and $(1 - p_t)$ factors cancel out in the expression for $RSV$
- Weak estimate, but doesn’t disagree violently with expectation that query terms appear in many but not all relevant documents
- Combining this method with the earlier approximation for $\overline{p_t}$, the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting
- For short documents (titles or abstracts) in one-pass retrieval situations, this estimate can be quite satisfactory
History and summary of assumptions
History and summary of assumptions

- Among the oldest formal models in IR
Among the oldest formal models in IR

- Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities.
History and summary of assumptions

Among the oldest formal models in IR

- Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities.

- Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
History and summary of assumptions

- Among the oldest formal models in IR
  - Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities
- Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
  - Boolean representation of documents/queries/relevance
History and summary of assumptions

- Among the oldest formal models in IR
  - Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities

- Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
  - Boolean representation of documents/queries/relevance
  - Term independence
History and summary of assumptions

- Among the oldest formal models in IR
  - Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities

- Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
  - Boolean representation of documents/queries/relevance
  - Term independence
  - Out-of-query terms do not affect retrieval
History and summary of assumptions

Among the oldest formal models in IR
- Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities

Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
- Boolean representation of documents/queries/relevance
- Term independence
- Out-of-query terms do not affect retrieval
- Document relevance values are independent
How different are vector space and BIM?
How different are vector space and BIM?

- They are not that different.
How different are vector space and BIM?

- They are not that different.
- In either case you build an information retrieval scheme in the exact same way.
How different are vector space and BIM?

- They are not that different.
- In either case you build an information retrieval scheme in the exact same way.
- For probabilistic IR, at the end, you score queries not by cosine similarity and tf-idf in a vector space, but by a slightly different formula motivated by probability theory.
How different are vector space and BIM?

- They are not that different.
- In either case you build an information retrieval scheme in the exact same way.
- For probabilistic IR, at the end, you score queries not by cosine similarity and tf-idf in a vector space, but by a slightly different formula motivated by probability theory.
- Next: how to add term frequency and length normalization to the probabilistic model.
Okapi BM25: Overview

Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it’s nonbinary) and length normalization. BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts. For modern full-text search collections, a model should pay attention to term frequency and document length. BestMatch25 (a.k.a BM25 or Okapi) is sensitive to these quantities.

BM25 is one of the most widely used and robust retrieval models.
Okapi BM25: Overview

- Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it’s nonbinary) and length normalization.
Okapi BM25: Overview

- Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it’s nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.
Okapi BM25: Overview

- Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it’s nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.
- For modern full-text search collections, a model should pay attention to term frequency and document length.
Okapi BM25: Overview

- Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it’s nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.
- For modern full-text search collections, a model should pay attention to term frequency and document length.
- BestMatch25 (a.k.a BM25 or Okapi) is sensitive to these quantities.
Okapi BM25: Overview

- Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it’s nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.
- For modern full-text search collections, a model should pay attention to term frequency and document length.
- BestMatch25 (a.k.a BM25 or Okapi) is sensitive to these quantities.
- BM25 is one of the most widely used and robust retrieval models.
Okapi BM25: Starting point

The simplest score for document \(d\) is just idf weighting of the query terms present in the document:

\[
\text{RSV}_d = \sum_{t \in q} \log \frac{N}{n_t}
\]
Okapi BM25: Starting point

- The simplest score for document \( d \) is just idf weighting of the query terms present in the document:
Okapi BM25: Starting point

- The simplest score for document $d$ is just idf weighting of the query terms present in the document:
The simplest score for document $d$ is just idf weighting of the query terms present in the document:

$$RSV_d = \sum_{t \in q} \log \frac{N}{n}$$
Okapi BM25 Basic Weighting

\[ RSV_d = \sum_{t \in q} \log \left( \frac{N}{n} \right) \cdot \left( k_1 + 1 \right) \frac{tf}{td} \left( 1 - b + b \times \frac{L_d}{L_{ave}} \right) + \frac{tf}{td} \]

- \( tf \): term frequency in document \( d \)
- \( L_d \): length of document \( d \)
- \( L_{ave} \): average document length in the whole collection
- \( k_1 \): tuning parameter controlling the document term frequency scaling
- \( b \): tuning parameter controlling the scaling by document length

Basic Weighting

- Improve idf term \( \log N/n \) by factoring in term frequency and document length.
Okapi BM25 Basic Weighting

- Improve idf term \([\log \frac{N}{n}]\) by factoring in term frequency and document length.

\[
RSV_d = \sum_{t \in q} \log \left[ \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}
\]

- \(N\): number of documents
- \(n\): number of documents containing term \(t\)
- \(k_1\): tuning parameter controlling the document term frequency scaling
- \(b\): tuning parameter controlling the scaling by document length
- \(L_d\): length of document \(d\)
- \(L_{ave}\): average document length in the whole collection
Okapi BM25 Basic Weighting

- Improve idf term \([\log N/n]\) by factoring in term frequency and document length.

\[
RSV_d = \sum_{t \in q} \log \left[ \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave}))) + tf_{td}}
\]

- \(tf_{td}\): term frequency in document \(d\)
Okapi BM25 Basic Weighting

- Improve idf term \([\log N/n]\) by factoring in term frequency and document length.

\[
RSV_d = \sum_{t \in q} \log \left[ \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1 ((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}
\]

- \(tf_{td}\): term frequency in document \(d\)
- \(L_d\ (L_{ave})\): length of document \(d\) (average document length in the whole collection)
Okapi BM25 Basic Weighting

- Improve idf term \([\log \frac{N}{n}]\) by factoring in term frequency and document length.

\[
RSV_d = \sum_{t \in q} \log \left[ \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}
\]

- \(tf_{td}\): term frequency in document \(d\)
- \(L_d\) (\(L_{ave}\)): length of document \(d\) (average document length in the whole collection)
- \(k_1\): tuning parameter controlling the document term frequency scaling
Okapi BM25 Basic Weighting

- Improve idf term $[\log N/n]$ by factoring in term frequency and document length.

$$RSV_d = \sum_{t \in q} \log \left[ \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}$$

- $tf_{td}$: term frequency in document $d$
- $L_d$ ($L_{ave}$): length of document $d$ (average document length in the whole collection)
- $k_1$: tuning parameter controlling the document term frequency scaling
- $b$: tuning parameter controlling the scaling by document length
Okapi BM25 weighting for Long queries

\[ d = \sum_{t \in q} \left[ \log \frac{N}{n_t} \right] \cdot (k_1 + 1) \cdot tf_t \cdot td + \left( k_3 + 1 \right) \cdot tf_t \cdot tq \]

\( tf_t \): term frequency in the query
\( k_3 \): tuning parameter controlling term frequency scaling of the query

No length normalization of queries (because retrieval is being done with respect to a single fixed query)

The above tuning parameters should ideally be set to optimize performance on a development test collection. In the absence of such optimization, experiments have shown reasonable values are to set \( k_1 \) and \( k_3 \) to a value between 1.2 and 2 and \( b = 0 \).
Okapi BM25 weighting for Long queries

- For long queries, use similar weighting for query terms

\[
RSV_d = \sum_{t \in q} \left[ \log \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}} \cdot \frac{(k_3 + 1)tf_{tq}}{k_3 + tf_{tq}}
\]
Okapi BM25 weighting for Long queries

For long queries, use similar weighting for query terms

\[ RSV_d = \sum_{t \in q} \left[ \log \frac{N}{n} \right] \cdot \frac{(k_1 + 1) tf_{td}}{k_1 ((1 - b) + b \times (L_d / L_{ave})) + tf_{td}} \cdot \frac{(k_3 + 1) tf_{tq}}{k_3 + tf_{tq}} \]

- \( tf_{tq} \): term frequency in the query \( q \)
Okapi BM25 weighting for Long queries

For long queries, use similar weighting for query terms

\[ RSV_d = \sum_{t \in q} \left[ \log \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave}))} + \frac{(k_3 + 1)tf_{tq}}{k_3 + tf_{tq}} \]

- \( tf_{tq} \): term frequency in the query \( q \)
- \( k_3 \): tuning parameter controlling term frequency scaling of the query
Okapi BM25 weighting for Long queries

- For long queries, use similar weighting for query terms

\[
RSV_d = \sum_{t \in q} \left[ \log \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}} \cdot \frac{(k_3 + 1)tf_{tq}}{k_3 + tf_{tq}}
\]

- \(tf_{tq}\): term frequency in the query \(q\)
- \(k_3\): tuning parameter controlling term frequency scaling of the query
- No length normalization of queries (because retrieval is being done with respect to a single fixed query)
Okapi BM25 weighting for Long queries

- For long queries, use similar weighting for query terms

\[
RSV_d = \sum_{t \in q} \left[ \log \frac{N}{n} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}} \cdot \frac{(k_3 + 1)tf_{tq}}{k_3 + tf_{tq}}
\]

- \(tf_{tq}\): term frequency in the query \(q\)
- \(k_3\): tuning parameter controlling term frequency scaling of the query
- No length normalization of queries (because retrieval is being done with respect to a single fixed query)
- The above tuning parameters should ideally be set to optimize performance on a development test collection. In the absence of such optimization, experiments have shown reasonable values are to set \(k_1\) and \(k_3\) to a value between 1.2 and 2 and \(b = 0.75\).
Okapi at TREC 7*

- All the TREC-7 searches used varieties of Okapi BM25, such as:

\[
\sum_{t \in q} \left[ \log \frac{N}{n} \right] \cdot \frac{(k_1 + 1) tf_{td}}{K + tf_{td}} \cdot \frac{(k_3 + 1) tf_{tq}}{k_3 + tf_{tq}} + k_2|Q| \frac{L_{ave} - L_d}{L_{ave} + L_d}
\]

- Where \(K = k_1((1 - b) + b \times \frac{L_d}{L_{ave}})\)
- \(k_2\) is a parameter depending on the nature of the query and possibly on the database
- \(k_2\) was always zero in all searches of TREC-7, simplifying the equation to:

\[
\sum_{t \in q} w^1 \cdot \frac{(k_1 + 1) tf_{td}}{K + tf_{td}} \cdot \frac{(k_3 + 1) tf_{tq}}{k_3 + tf_{tq}}
\]

- Where \(w^1 = \left[ \log \frac{N}{n} \right]\), aka Roberstons-Jones weight
  (Roberston et.al.,
Outline

1. Inception
2. Probabilistic Approach to IR
3. Data
4. Basic Probability Theory
5. Probability Ranking Principle
6. Extensions to BIM: Okapi
7. Performance measure
8. Comparision of Models
Performance measures used
Outline

1. Inception
2. Probabilistic Approach to IR
3. Data
4. Basic Probability Theory
5. Probability Ranking Principle
6. Extensions to BIM: Okapi
7. Performance measure
8. Comparision of Models
Probabilistic model vs. other models

- Probabilistic model vs. other models
  - Boolean model
  - Probabilistic models support ranking and thus are better than the simple Boolean model.
  - Vector space model
  - The vector space model is also a formally defined model that supports ranking.
  - Why would we want to look for an alternative to the vector space model?
Probabilistic model vs. other models

- Boolean model
Probabilistic model vs. other models

- **Boolean model**
  - Probabilistic models support ranking and thus are better than the simple Boolean model.
Probabilistic model vs. other models

- **Boolean model**
  - Probabilistic models support ranking and thus are better than the simple Boolean model.

- **Vector space model**
Probabilistic model vs. other models

- **Boolean model**
  - Probabilistic models support ranking and thus are better than the simple Boolean model.

- **Vector space model**
  - The vector space model is also a formally defined model that supports ranking.
Probabilistic model vs. other models

- Boolean model
  - Probabilistic models support ranking and thus are better than the simple Boolean model.

- Vector space model
  - The vector space model is also a formally defined model that supports ranking.
  - Why would we want to look for an alternative to the vector space model?
Probabilistic vs. vector space model
Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.
Vector space model: rank documents according to similarity to query.

The notion of similarity does not translate directly into an assessment of “is the document a good document to give to the user or not?”
Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.
- The notion of similarity does not translate directly into an assessment of “is the document a good document to give to the user or not?”
- The most similar document can be highly relevant or completely nonrelevant.
Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.
- The notion of similarity does not translate directly into an assessment of “is the document a good document to give to the user or not?”
- The most similar document can be highly relevant or completely nonrelevant.
- Probability theory is arguably a cleaner formalization of what we really want an IR system to do: give relevant documents to the user.
Which ranking model should I use?

- If you want something basic and simple, use vector space with tf-idf weighting.
- If you want a state-of-the-art ranking model with excellent performance, use language models or BM25 with tuned parameters.
- In between: BM25 or language models with no or just one tuned parameter.
Which ranking model should I use?

- I want something basic and simple → use vector space with tf-idf weighting.
Which ranking model should I use?

- I want something basic and simple → use vector space with tf-idf weighting.
- I want to use a state-of-the-art ranking model with excellent performance → use language models or BM25 with tuned parameters.
Which ranking model should I use?

- I want something basic and simple → use vector space with tf-idf weighting.
- I want to use a state-of-the-art ranking model with excellent performance → use language models or BM25 with tuned parameters
- In between: BM25 or language models with no or just one tuned parameter
Resources

- Chapter 11 of IIR
- Resources at http://cislmu.org
- Presentation on "Probabilistic Information Retrieval" by Dr. Suman Mitra at: http://www.irs.res.in/winter-school/slides/ProbabilisticModel.ppt