**Motivation and Outline**

- Background
- Definitions, etc.
- The Problem
- 100,000+ pages
- The Solution
- Ranking docs
- Vector space
- Extensions
- Relevance feedback
- Clustering
- Query expansion, etc.

**Motivation**

- IR: representation, storage, organization of, and access to information items
- Focus is on the user information need
- User information need:
  - Find all docs containing information on college tennis teams which:
    1. are maintained by a USA university and
    2. participate in the NCAA tournament.
- Emphasis is on the retrieval of information (not data)

**Motivation**

- Data retrieval
  - which docs contain a set of keywords?
  - Well defined semantics
  - a single erroneous object implies failure!
- Information retrieval
  - information about a subject or topic
  - semantics is frequently loose
  - small errors are tolerated
- IR system:
  - interpret contents of information items
  - generate a ranking which reflects relevance
  - notion of relevance is most important

**Measuring Performance**

- Precision
  - Proportion of selected items that are correct
  \[
  \text{Precision} = \frac{TP}{TP + FP}
  \]
- Recall
  - Proportion of target items that were selected
  \[
  \text{Recall} = \frac{TP}{TP + FN}
  \]
- Precision-Recall curve
  - Shows tradeoff

**Basic Concepts**

- The User Task
  - Retrieval
    - information or data
    - purposeful
  - Browsing
    - glancing around
    - F1: cars, Le Mans, France, tourism
Basic Concepts

- Logical view of the documents
  - Document representation viewed as a continuum: logical view of docs might shift

The Retrieval Process

Introduction

- IR systems usually adopt index terms to process queries
- Index term:
  - a keyword or group of selected words
  - any word (more general)
- Stemming might be used:
  - connect: connecting, connection, connections
- An inverted file is built for the chosen index terms

Introduction

- Matching at index term level is quite imprecise
- No surprise that users get frequently unsatisfied
- Since most users have no training in query formation, problem is even worst
- Frequent dissatisfaction of Web users
- Issue of deciding relevance is critical for IR systems: ranking

Introduction

- A ranking is an ordering of the documents retrieved that (hopefully) reflects the relevance of the documents to the user query
- A ranking is based on fundamental premisses regarding the notion of relevance, such as:
  - common sets of index terms
  - sharing of weighted terms
  - likelihood of relevance
- Each set of premisses leads to a distinct IR model
**Retrieval: Ad Hoc x Filtering**

- Filtering:
  - **User 2 Profile**
  - **User 1 Profile**

- Documents Stream
  - Docs Filtered for User 2
  - Docs for User 1

**Classic IR Models - Basic Concepts**

- Not all terms are equally useful for representing the document contents: less frequent terms allow identifying a narrower set of documents
- The importance of the index terms is represented by weights associated to them
- Let
  - $k_i$ be an index term
  - $d_j$ be a document
  - $w_{ij}$ is a weight associated with $(k_i,d_j)$
- The weight $w_{ij}$ quantifies the importance of the index term for describing the document contents
The Boolean Model

- Simple model based on set theory
- Queries specified as boolean expressions
  - precise semantics
  - neat formalism
  - $q = ka \land (kb \lor -kc)$
- Terms are either present or absent. Thus, $wij \in \{0,1\}$
- Consider
  - $q = ka \land (kb \lor -kc)$
  - vec(qdnf) = (1,1,1) \lor (1,1,0) \lor (1,0,0)
  - vec(qcc) = (1,1,0) is a conjunctive component

Drawbacks of the Boolean Model

- Retrieval based on binary decision criteria with no notion of partial matching
- No ranking of the documents is provided (absence of a grading scale)
- Information need has to be translated into a Boolean expression which most users find awkward
- The Boolean queries formulated by the users are most often too simplistic
- As a consequence, the Boolean model frequently returns either too few or too many documents in response to a user query

The Vector Model

- Use of binary weights is too limiting
- Non-binary weights provide consideration for partial matches
- These term weights are used to compute a degree of similarity between a query and each document
- Ranked set of documents provides for better matching

The Vector Model

- Define:
  - $wij > 0$ whenever $ki \in dj$
  - $wiq \geq 0$ associated with the pair $(ki,q)$
  - vec($dj$) = (w1j, w2j, ..., wtj)
  - vec(q) = (w1q, w2q, ..., wtq)
  - To each term $ki$ is associated a unitary vector vec($i$)
  - The unitary vectors vec($i$) and vec($j$) are assumed to be orthonormal (i.e., index terms are assumed to occur independently within the documents)
- The $t$ unitary vectors vec($i$) form an orthonormal basis for a $t$-dimensional space
- In this space, queries and documents are represented as weighted vectors

Document Vectors

- Documents are represented as “bags of words”
- Represented as vectors when used computationally
  - A vector is like an array of floating point
  - Has direction and magnitude
  - Each vector holds a place for every term in the collection
- Therefore, most vectors are sparse
Vector Space Example

- a. System and human system engineering testing of EPS
- b. A survey of user opinion of computer system response time
- c. The EPS user interface management system
- d. Human machine interface for ABC computer applications
- e. Relation of user perceived response time to error measurement
- f. The generation of random, binary, ordered trees
- g. The intersection graph of paths in trees
- h. Graph minors IV: Widths of trees and well-quasi-ordering
- i. Graph minors: A survey

Documents in 3D Space

- Interface 0 1 0 0 0 0 0 0 0
- User 0 0 1 0 0 0 0 0 0
- System 0 0 0 1 0 0 0 0 0
- Human 0 0 0 0 1 0 0 0 0
- Computer 0 0 0 0 0 1 0 0 0
- Response 0 0 0 0 0 0 1 0 0
- Time 0 0 0 0 0 0 0 1 0
- EPS 0 0 0 0 0 0 0 0 1
- Survey 0 0 0 0 0 0 0 0 0
- Trees 0 0 0 0 0 0 0 0 0
- Graph 0 0 0 0 0 0 0 0 0
- Minors 0 0 0 0 0 0 0 0 0

We Can Plot the Vectors

\[ \text{Sim}(q, dj) = \cos(\Theta) = \frac{\| \text{vec}(dj) \cdot \text{vec}(q) \|}{|dj| \cdot |q|} = \frac{\sum wij \cdot wiq}{|dj| \cdot |q|} \]

Since \( wij > 0 \) and \( wiq > 0 \),
0 <= sim(q,dj) <= 1

A document is retrieved even if it matches the query terms only partially.

Vector Space Example cont.

\[ \cos(\Theta_{ij}) = \frac{A \cdot B}{|A| \cdot |B|} \]

Answering a Query Using Vector Space

- Represent query as vector
- Compute distances to all documents
- Rank according to distance
- Example
  - “computer system”
The Vector Model

- \( \text{Sim}(q,d) = [\Sigma \ w_{ij} \cdot \ w_{iq}] / |d| \cdot |q| \)
- How to compute the weights \( w_{ij} \) and \( w_{iq} \)?
  - Simple keyword frequencies tend to favor common words
  - E.g. Query: The Computer Tomography
- A good weight must take into account two effects:
  - quantification of intra-document contents (similarity)
    - \( tf \) factor, the term frequency within a document
  - quantification of inter-documents separation (dissimilarity)
    - \( idf \) factor, the inverse document frequency
  - \( w_{ij} = tf(i,j) \cdot idf(i) \)

- Let,
  - \( N \) be the total number of docs in the collection
  - \( n_i \) be the number of docs which contain \( ki \)
  - \( \text{freq}(i,j) \), raw frequency of \( ki \) within \( dj \)
- A normalized \( tf \) factor is given by
  - \( f(i,j) = \frac{\text{freq}(i,j)}{\max(\text{freq}(l,j))} \)
  - where the maximum is computed over all terms which occur within the document \( dj \)
- The \( idf \) factor is computed as
  - \( idf(i) = \log \left( \frac{N}{n_i} \right) \)
  - the log is used to make the values of \( tf \) and \( idf \) comparable. It can also be interpreted as the amount of information associated with the term \( ki \).

- The best term-weighting schemes use weights which are given by
  - \( w_{ij} = f(i,j) \cdot \log(N/n_i) \)
  - the strategy is called a \( tf-idf \) weighting scheme
- For the query term weights, a suggestion is
  - \( w_{iq} = (0.5 + \left(0.5 \cdot \frac{\text{freq}(i,q)}{\max(\text{freq}(l,q))}\right) \cdot \log(N/n_i) \)
- The vector model with \( tf-idf \) weights is a good ranking strategy with general collections
- The vector model is usually as good as the known ranking alternatives. It is also simple and fast to compute.

Advantages:
- term-weighting improves quality of the answer set
- partial matching allows retrieval of docs that approximate the query conditions
- cosine ranking formula sorts documents according to degree of similarity to the query

Disadvantages:
- assumes independence of index terms (??); not clear that this is bad though

The Vector Model: Example 1

<table>
<thead>
<tr>
<th></th>
<th>k1</th>
<th>k2</th>
<th>k3</th>
<th>q ( \cdot dj )</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>d2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d3</td>
<td>1</td>
<td>1</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>d4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>d6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>d7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>q</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Probabilistic Model**

- Objective: to capture the IR problem using a probabilistic framework
- Given a user query, there is an ideal answer set
- Querying as specification of the properties of this ideal answer set (clustering)
- But, what are these properties?
- Guess at the beginning what they could be (i.e., guess initial description of ideal answer set)
- Improve by iteration

**Probabilistic Ranking Principle**

- Given a user query \( q \) and a document \( d_j \), the probabilistic model tries to estimate the probability that the user will find the document \( d_j \) interesting (i.e., relevant). The model assumes that this probability of relevance depends on the query and the document representations only. Ideal answer set is referred to as \( R \) and should maximize the probability of relevance. Documents in the set \( R \) are predicted to be relevant.
- But,
  - how to compute probabilities?
  - what is the sample space?

**The Ranking**

- Probabilistic ranking computed as:
  - \( \text{sim}(q,d_j) = \frac{P(R | \text{vec}(d_j))}{P(\neg R | \text{vec}(d_j))} \)
  - This is the odds of the document \( d_j \) being relevant
  - Taking the odds minimize the probability of an erroneous judgement
- Definition:
  - \( w_{ij} \in \{0,1\} \)
  - \( P(R | \text{vec}(d_j)) \) : probability that given doc is relevant
  - \( P(\neg R | \text{vec}(d_j)) \) : probability doc is not relevant

\[
\text{sim}(d_j,q) = \frac{P(R | \text{vec}(d_j))}{P(\neg R | \text{vec}(d_j))} = \frac{\prod P(k_i | R)}{\prod P(\neg k_i | R)} \frac{\prod P(k_i | \neg R)}{\prod P(\neg k_i | \neg R)}
\]
The Ranking

\[ \text{sim}(d_j, q) \sim \log \left[ \prod P(k_i | R) \right] \cdot \log \left[ \prod P(\neg k_i | \neg R) \right] \]

\[ \sim K \cdot \log \left[ \frac{\prod P(k_i | R)}{\prod P(\neg k_i | \neg R)} \right] \]

\[ \sim \sum w_i q \cdot w_j \cdot \left( \log \frac{P(k_i | R)}{P(\neg k_i | \neg R)} + \log \frac{P(\neg k_i | R)}{P(k_i | \neg R)} \right) \]

where

\[ P(\neg k_i | R) = 1 - P(k_i | R) \]

\[ P(\neg k_i | \neg R) = 1 - P(k_i | \neg R) \]

The Initial Ranking

\[ \text{sim}(d_j, q) \sim \sum w_i q \cdot w_j \cdot \left( \log \frac{P(k_i | R)}{P(\neg k_i | \neg R)} + \log \frac{P(\neg k_i | R)}{P(k_i | \neg R)} \right) \]

\[ \text{Probabilities } P(k_i | R) \text{ and } P(k_i | \neg R) ? \]

\[ \text{Estimates based on assumptions:} \]

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

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

where \( N_i \) is the number of docs that contain \( k_i \)

\[ \text{Use this initial guess to retrieve an initial ranking} \]

\[ \text{Improve upon this initial ranking} \]

Improving the Initial Ranking

\[ \text{sim}(d_j, q) \sim \sum w_i q \cdot w_j \cdot \left( \log \frac{P(k_i | R)}{P(\neg k_i | \neg R)} + \log \frac{P(\neg k_i | R)}{P(k_i | \neg R)} \right) \]

\[ \text{Let} \]

\[ V : \text{set of docs initially retrieved} \]

\[ V_i : \text{subset of docs retrieved that contain } k_i \]

\[ \text{Reevaluate estimates:} \]

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

\[ P(k_i | \neg R) = \frac{N_i - V_i}{N - V} \]

\[ \text{Repeat recursively} \]

To avoid problems with \( V=1 \) and \( V_i=0 \):

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

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

Also,

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

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

Pluses and Minuses

\[ \text{Advantages:} \]

\[ \text{Docs ranked in decreasing order of probability of relevance} \]

\[ \text{Disadvantages:} \]

\[ \text{need to guess initial estimates for } P(k_i | R) \]

\[ \text{method does not take into account } \text{tf} \text{ and } \text{idf} \text{ factors} \]

Brief Comparison of Classic Models

\[ \text{Boolean model does not provide for partial matches} \]

\[ \text{and is considered to be the weakest classic model} \]

\[ \text{Salton and Buckley did a series of experiments that} \]

\[ \text{indicate that, in general, the vector model outperforms} \]

\[ \text{the probabilistic model with general collections} \]

\[ \text{This seems also to be the view of the research} \]

\[ \text{community} \]