Content of this Lecture

- IR Models
- Boolean Model
- Vector Space Model
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
- Other IR Models
Information Retrieval Core

• The core question in IR: Which from a given set of (normalized) documents are relevant for a given query?
• Ranking: How relevant for a given query is each document?
Judging Relevance

- **Retrieval**: Adhoc Filtering
  - Classic Models
    - Boolean
    - Vector-Space
    - Probabilistic
  - Structured Models
    - Non-Overlapping Lists
    - Proximal Nodes
  - Probabilistic
    - Inference Network
    - Belief Network

- **Algebraic**
  - Generalized Vector
  - Lat. Semantic Index
  - Neural Networks

- **Set Theoretic**
  - Fuzzy
  - Extended Boolean

- **Browsing**
  - Flat
  - Structure Guided Hypertext

[BYRN99]
Notation

- Most models we discuss use the "Bag of Words" view
- Definition
  - Let $D$ be the set of all normalized documents, $d \in D$ is a document
  - Let $K$ be the set of all unique tokens in $D$, $k \in K$ is a token
    - Can as well be terms
  - Let $w$ be the function that maps a given $d$ to its bag of tokens from $K$ (its bag-of-words)
  - Let $v_d$ by a vector of size $|K|$ with
    - $v_d[i] = 0 \text{ iff } k_i \notin w(d)$
    - $v_d[i] = 1 \text{ iff } k_i \in w(d)$
- Note: Later, we will use weights instead of a Boolean membership function
Content of this Lecture

- IR Models
- **Boolean Model**
- Vector Space Model
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
- Other IR Models
Boolean Model

• Simple relevance model based on set theory
• Queries are specified as **Boolean expressions** over tokens
  - Tokens are atoms
  - Atoms are connected by AND, OR, NOT, (XOR, ...)
  - Parenthesis as usual (but ignored here)
• Relevance of a document
  - Let q contain the atoms \(<k_1, k_2, ...>\)
  - An **atom** \(k_i\) evaluates to true for \(d\) iff \(v_d[k_i] = 1\)
  - Compute values of all atoms for each \(d\)
  - Compute value of q for \(d\) as **logical expression** over atoms
  - Result is **true or false**
Properties

• Simple, clear semantics, widely used in (early) systems

• Disadvantages
  – No partial matching
    • Suppose query \( k_1 \land k_2 \land \ldots \land k_9 \)
    • A doc \( d \) with \( k_1 \land k_2 \land \ldots k_8 \) is as irrelevant as one with none of the terms
  – No ranking
  – Token cannot be weighted
    • But some are more important for a doc than others
  – Average users don’t like (understand) Boolean expressions

• Often unsatisfactory results especially for non IR-experts
  – Too many documents (too few restrictions, many OR)
  – Too few documents (too many restrictions, many AND)
  – Several extensions exist
A Note on Implementation

• One should not iterate over D, but use a term index
  - Assume we have an index with fast operation find: K→PD
  - Search each atom k_i of the query, resulting in a set D_i⊆D
  - Evaluate query in the given order using set operations on D_i’s
    • k_i ∧ k_j : D_i ∩ D_j
    • k_i ∨ k_j : D_i ∪ D_j
    • NOT k_i : D\D_i

• Improvements: Cost-based evaluation
  - Evaluate sub-expressions first that result in smaller intermediate results
  - Less memory requirements, faster intersections, …
Negation in the Boolean Model

- Evaluating “**NOT** $k_i$” can be very expensive
  - If $k_i$ is not a stop word, result is very large: $|D \setminus D_i| \approx |D|$
    - Most terms appear in almost no documents
    - Recall Zipf’s Law – the tail of the distribution

- **Solution 1:** Disallow negation
  - This is what many web search engines do

- **Solution 2:** Allow only in the form “$k_i \land \text{NOT} \ k_j$”
  - Should not use implementation scheme as given before
    - $D_{\text{not-kj}}$ would be very large
  - Better: $D := D_i \setminus D_j$
Content of this Lecture

- IR Models
- Boolean Model
- **Vector Space Model**
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
- Other IR Models
Vector Space Model

  - A breakthrough in IR
  - Still most popular model today

- General idea
  - Fix a vocabulary $K$
  - View each doc and query as a point in a $|K|$-dimensional space
  - Rank docs according to distance from the query in that space

- Main advantages
  - Natural ranking of docs (according to distance)
  - Naturally supports partial matching (increases distance)
Vector Space

- Each term is one dimension
  - Different suggestions for determining co-ordinates, i.e., term weights
- The closest docs are the most relevant ones
  - Rationale: Vectors correspond to themes which are loosely related to sets/bags of terms
  - Distance between vectors ~ distance between themes
  - Different suggestions for defining distance
The Angle between Two Vectors

- Recall: The **scalar product** between two vectors $v$ and $w$ of equal dimension is defined as

\[ v \cdot w = |v| \times |w| \times \cos(v, w) \]

- This gives us the angle

\[ \cos(v, w) = \frac{v \cdot w}{|v| \times |w|} \]

  - With

\[ |v| = \sqrt{\sum v_i^2} \quad \text{and} \quad v \cdot w = \sum v_i \times w_i \]
Distance as Angle

Distance = cosine of the angle between doc d and query q

\[ \text{sim}(d, q) = \cos(v_d, v_q) = \frac{v_d \cdot v_q}{|v_d| \times |v_q|} = \frac{\sum(v_{d[i]} \times v_{q[i]})}{\sqrt{\sum v_{d[i]}^2} \times \sqrt{\sum v_{q[i]}^2}} \]

Length normalization

Can be dropped for ranking
Example

- Assume stop word removal, stemming, and **binary weights**

<table>
<thead>
<tr>
<th>Text</th>
<th>verkauf</th>
<th>haus</th>
<th>italien</th>
<th>gart</th>
<th>miet</th>
<th>blüh</th>
<th>woll</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Wir verkaufen Häuser in Italien</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Häuser mit Gärten zu vermieten</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Häuser: In Italien, um Italien, um Italien herum</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Die italienschen Gärtner sind im Garten</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Der Garten in unserem italienschen Haus blüht</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Q Wir wollen ein Haus mit Garten in Italien mieten</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Ranking

\[
sim(d, q) = \frac{\sum (v_q[i] \times v_d[i])}{\sqrt{\sum v_d[i]^2}}
\]

- \(\sim \frac{(1 \times 0 + 1 \times 1 + 1 \times 1 + 0 \times 1 + 0 \times 1 + 0 \times 0 + 0 \times 1)}{\sqrt{3}} \sim 1.15\)
- \(\sim \frac{(1 + 1 + 1)}{\sqrt{3}} \sim 1.73\)
- \(\sim \frac{(1 + 1)}{\sqrt{2}} \sim 1.41\)
- \(\sim \frac{(1 + 1)}{\sqrt{2}} \sim 1.41\)
- \(\sim \frac{(1 + 1 + 1)}{\sqrt{4}} \sim 1.5\)

<table>
<thead>
<tr>
<th>Rg</th>
<th>Q: Wir wollen ein Haus mit Garten in Italien mieten</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>d2: Häuser mit Gärten zu vermieten</td>
</tr>
<tr>
<td>2</td>
<td>d5: Der Garten in unserem italienschen Haus blüht</td>
</tr>
<tr>
<td>3</td>
<td>d4: Die italienschen Gärtner sind im Garten</td>
</tr>
<tr>
<td>4</td>
<td>d3: Häuser: In Italien, um Italien, um Italien herum</td>
</tr>
<tr>
<td>5</td>
<td>d1: Wir verkaufen Häuser in Italien</td>
</tr>
</tbody>
</table>
Introducing Term Weights

• Definition

Let $D$ be a document collection, $K$ be the set of all terms in $D$, $d \in D$ and $k \in K$

- The term frequency $t_{dk}$ is the frequency of $k$ in $d$
- The document frequency $d_{f,k}$ is the frequency of docs in $D$ containing $k$
  • This should rather be called “corpus frequency”
  • Sometimes defined as the frequency of occurrences of $k$ in $D$
  • Both definitions are valid and both are used
- The inverse document frequency $id_{f,k}$ is $id_{f,k} = |D| / d_{f,k}$
  • In practice, one usually uses $id_{f,k} = \log(|D| / d_{f,k})$
Ranking with TF scoring

$$\text{sim}(d, q) = \frac{\sum (v_q[i] * v_d[i])}{\sqrt{\sum v_d[i]^2}}$$

- $\text{sim}(d_1, q) = (1*0+1*1+1*1+0*1+0*1+0*0+0*1) / \sqrt{3} \sim 1.15$
- $\text{sim}(d_2, q) = (1+1+1) / \sqrt{3} \sim 1.73$
- $\text{sim}(d_3, q) = (1+3) / \sqrt{10} \sim 1.26$
- $\text{sim}(d_4, q) = (1+2) / \sqrt{5} \sim 1.34$
- $\text{sim}(d_5, q) = (1+1+1) / \sqrt{4} \sim 1.5$

<table>
<thead>
<tr>
<th>Rg</th>
<th>Q: Wir wollen ein <strong>Haus</strong> mit <strong>Garten</strong> in <strong>Italien</strong> mieten</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$d_2$: <strong>Häuser</strong> mit <strong>Gärten</strong> zu vermieten</td>
</tr>
<tr>
<td>2</td>
<td>$d_5$: Der <strong>Garten</strong> in unserem <strong>italienschen Haus</strong> blüht</td>
</tr>
<tr>
<td>3</td>
<td>$d_4$: Die <strong>italienschen Gärtner</strong> sind im <strong>Garten</strong></td>
</tr>
<tr>
<td>4</td>
<td>$d_3$: <strong>Häuser</strong>: In <strong>Italien</strong>, um <strong>Italien</strong>, um <strong>Italien</strong> herum</td>
</tr>
<tr>
<td>5</td>
<td>$d_1$: Wir verkaufen <strong>Häuser</strong> in <strong>Italien</strong></td>
</tr>
</tbody>
</table>
Alternative Scoring: TF*IDF

• 1st problem: The longer a doc, the higher the probability of finding query terms by pure chance
  – Solution: Normalize TF values on document length (yields $0 \leq w_{dk} \leq 1$)
    \[
    tf_{dk} = \frac{tf_{dk}}{|d|} = \frac{tf_{dk}}{\sum_{j=1}^{k} tf_{dj}}
    \]
  – Note: Longer docs also get down-ranked by normalization on doc-length in similarity function. Use only one measure!

• 2nd problem: Terms frequent in D don’t help to discriminate and should be scored less
  \[
  v_d[k] = tf_{dk} \times idf_k
  \]
**Example TF*IDF**

<table>
<thead>
<tr>
<th></th>
<th>IDF</th>
<th>5</th>
<th>5/4</th>
<th>5/4</th>
<th>5/3</th>
<th>5</th>
<th>5</th>
<th>DIV-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (tf)</td>
<td></td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 (tf)</td>
<td></td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 (tf)</td>
<td></td>
<td>1/4</td>
<td>3/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 (tf)</td>
<td></td>
<td></td>
<td>1/3</td>
<td>2/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 (tf)</td>
<td></td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- \(\text{sim}(d_1,q) = (5/4 \times 1/3 + 5/4 \times 1/3) / \sqrt{3.13} \approx 1.51\)
- \(\text{sim}(d_2,q) = (5/4 \times 1/3 + 5/3 \times 1/3 + 5 \times 1/3) / \sqrt{3.26} \approx 4.80\)
- \(\text{sim}(d_3,q) = (5/4 \times 1/4 + 5/4 \times 3/4) / \sqrt{0.98} \approx 1.57\)
- \(\text{sim}(d_4,q) = (5/4 \times 1/3 + 5/3 \times 2/3) / \sqrt{1.41} \approx 2.08\)
- \(\text{sim}(d_5,q) = (5/4 \times 1/4 + 5/4 \times 1/4 + 5/3 \times 1/4) / \sqrt{1.93} \approx 2.08\)

<table>
<thead>
<tr>
<th>wollen ein <strong>Haus</strong> mit <strong>Garten</strong> in <strong>Italien</strong> mieten</th>
<th>wollen ein <strong>Haus</strong> mit <strong>Garten</strong> in <strong>Italien</strong> mieten</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>d2</strong>: <strong>Häuser mit Gärten zu vermieten</strong></td>
<td><strong>Häuser mit Gärten zu vermieten</strong></td>
</tr>
<tr>
<td><strong>d5</strong>: Der <strong>Garten</strong> in unserem <strong>italienschen Haus</strong> blüht</td>
<td>Der <strong>Garten</strong> in unserem <strong>italienschen Haus</strong> blüht</td>
</tr>
<tr>
<td><strong>d4</strong>: Die <strong>italienschen Gärtner</strong> sind im <strong>Garten</strong></td>
<td>Die <strong>italienschen Gärtner</strong> sind im <strong>Garten</strong></td>
</tr>
<tr>
<td><strong>d3</strong>: <strong>Häuser</strong>: In <strong>Italien</strong>, um <strong>Italien</strong>, um <strong>Italien</strong> herum</td>
<td><strong>Häuser</strong>: In <strong>Italien</strong>, um <strong>Italien</strong>, um <strong>Italien</strong> herum</td>
</tr>
<tr>
<td><strong>d1</strong>: Wir verkaufen <strong>Häuser</strong> in <strong>Italien</strong></td>
<td>Wir verkaufen <strong>Häuser</strong> in <strong>Italien</strong></td>
</tr>
</tbody>
</table>
TF*IDF in Short

- Give query terms in a doc $d$ high weights which are (1) frequent in $d$ and (2) infrequent in $D$
- IDF deals with the consequences of Zipf’s law
  - The few very frequent (and unspecific) terms get lower scores
  - The many infrequent (and specific) terms get higher scores
- Interferes with stop word removal
  - If stop words are removed, IDF might not be necessary any more
  - If IDF is used, stop word removal might not be necessary any more
- Many variations: log? Smoothing?
A Concrete (and Popular) VSM-Model

• Okapi BM25
  - Okapi: First system which used it (80ties)
  - BM25: Best-Match, version 25 (roughly)

• Good results in several TREC evaluations

\[
sim(d,q) = \sum_{k \in q} IDF(k) \times \frac{tf_{dk} \times (k_1 + 1)}{tf_{dk} + k_1 \times \left(1 - b + b \times \frac{|d|}{a}\right)}; \quad IDF(k) = \frac{|D| - tf_k + 0.5}{tf_k + 0.5}
\]

- \(k_1, b\) constants (often \(b=0.75, k_1=0.2\))
- \(a\) is the average document length in \(D\)
Distance Measure

- Why not use Euclidean distance?
- Length of vectors would be much more important
- Since queries usually are very short, very short documents would always win
- Cosine measures normalizes by the length of both vectors
Shortcomings

• We assume that all terms are independent
  – Clearly wrong: some terms are semantically closer than others
    • Their co-appearance doesn’t mean more than only one appearance
    • The appearance of “red” in a doc with “wine” doesn’t mean much
  – Extension: Topic-based Vector Space Model (LSI - later)

• No treatment of synonyms (query expansion, …)

• No treatment of homonyms
  – Different senses = different dimensions
  – We would need to disambiguate terms into their senses (later)

• Term-order independent
  – But order carries semantic meaning (object? subject?)
Content of this Lecture

- IR Models
- Boolean Model
- Vector Space Model
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
Interactive IR

• Recall: IR is a process, not a single query
• Relevance feedback
  - User poses initial query
  - System computes ranked answer
  - User judges the relevance of the (top-k) results
  - System generates new (improved) ranked answers
    • User never needs to pose another query
    • New query is generated by the system
  - Loop until satisfaction
Relevance Feedback

• Basic assumptions
  - Relevant docs are similar to each other – the common theme should be emphasized
  - Irrelevant docs are different from relevant docs – the differences should be de-emphasized

• “Emphasize, de-emphasize” – Modify terms and weights
  - Query expansion: Add new terms to the query
    • From the relevant documents
    • More aggressive: add “NOT” with terms from irrelevant docs
  - Term re-weighting: Assign new weights to terms
    • Up-weight terms from the relevant docs
    • Down-weight terms from the irrelevant docs
Rocchio Algorithm

- Let $R$ ($N$) be the set of docs marked as relevant (irrelevant)
- Rocchio: Adapt query vector after each feedback

$$v_{q_{new}} = \alpha \cdot v_q + \beta \cdot \frac{1}{|R|} \sum_{d \in R} v_d - \gamma \cdot \frac{1}{|N|} \sum_{d \in N} v_d$$

- $\alpha$: Do not forget the original query
- Implicitly performs query expansion and term re-weighting

- How to choose $\alpha$, $\beta$, $\gamma$?
  - Tuning with gold standard sets – difficult
  - Educated guess followed by user studies
Example

Let $\alpha=0.5$, $\beta=0.5$, $\gamma=0$, $K=$\{information, science, retrieval, system\}

- $d_1$ = "information science" = $(0.8, 0.4, 0, 0)$
- $d_2$ = "retrieval systems" = $(0, 0, 0.8, 0.2)$
- $q$ = "retrieval information" = $(0.4, 0, 0.8, 0)$

If $d_1$ were marked as relevant
\[ q' = \frac{1}{2} q + \frac{1}{2} d_1 = (0.6, 0.2, 0.4, 0) \]

If $d_2$ were marked as relevant
\[ q'' = \frac{1}{2} q + \frac{1}{2} d_2 = (0.2, 0, 0.8, 0.1) \]
Choices for N

• How can we determine N?
  - Ask the user for explicit negative feedback
    • More work for the user
  - Use only relevant feedback and $N = D \setminus R$
    • Infeasible: $N$ too large and with low confidence
  - Implicit: Docs presented for assessment and not marked relevant
    • User hopefully looked at all suggestions
    • But most users look at only a few – low confidence

• Generally: Large $N$ make things very slow
  - Query vector after first round has $\sim |K|$ non-null values

• Problem: $R$ has a theme, $N$ probably very heterogeneous
  - High likelihood that terms get weights reflecting only the corpus, not the “not in $R$” property
Variations

- Alternative treatment for N
  - Intuition: Non-relevant docs are heterogeneous and tear in every direction – better to only take the worst instead of all of them

  \[ v_{q_{new}} = \alpha * v_q + \beta * \frac{1}{|R|} \sum_{d \in R} v_d - \gamma * \{v_d | d = \arg \min_{d \in N}(sim(v_q, v_d)) \} \]

  - But: Probably many documents with similarity 0 – which to take?
  - Engines are tuned to find most relevant docs – inefficient

- Probably most popular choice: Ignore N
Effects of Relevance Feedback

• Advantages
  – Improves results (many studies) compared to single queries
  – Comfortable? Users need not generate new queries themselves
  – Iterative process converging to the best possible answer
  – Especially helpful for increasing recall
    • Due to query expansion – kind-of synonym expansion

• Disadvantages
  – Still requires some work by the user
    • Excite: Only 4% used relevance feedback ("more of this" button)
  – Writing a new query based on returned results might be faster (and easier and more successful) than rating results
  – Assumes that relevant docs are similar
    • What if user searches for all meanings of “jaguar”?
Collaborative Filtering

- More inputs for improving IR performance
- **Collaborative filtering**: Return to the user what other yet similar users liked
  - “Customers who bought this book also bought …”
  - In IR: Find users posing similar queries and look at what they did with the answers
    - In e-Commerce: Which produces did they buy? (very reliable)
    - In IR, we need to approximate
      - Documents a user clicked on (if known)
      - Did the user look at the second page? (Low credit for first results)
      - Did the user pose a “refinement query” next?
      - ...
    - All these measures are not very reliable; we need many users
Thesaurus-based Query Expansion [M07, CS276]

- Expand query with synonyms and hyponyms of each term
  - feline → feline cat
  - One may weight added terms less than original query terms
- Often used in scientific IR systems (Medline)
- Requires high quality thesaurus
- General observation
  - Increases recall
  - May significantly decrease precision
    - “interest rate” → “interest rate fascinate evaluate”
    - Do synonyms really exist?
Self Assessment

• Explain the vector space model
• How is the size of $K$ (vocabulary) influenced by pre-processing?
• Describe some variations of deducing term weights
• How could we extend the VSM to also consider the order of terms (to a certain degree)?
• How does the Rocchio algorithm determine the next query after feedback?
• How can we determine a useful set of negative documents in relevance feedback?
• How does relevance feedback work in current search engines?