Content-based recommendation
Content-based recommendation

- While CF – methods do not require any information about the items,
  - it might be reasonable to exploit such information; and
  - recommend fantasy novels to people who liked fantasy novels in the past

- What do we need:
  - some information about the available items such as the genre ("content")
  - some sort of user profile describing what the user likes (the preferences)

- The task:
  - learn user preferences
  - locate/recommend items that are "similar" to the user preferences
What is the "content"?

- Most CB-recommendation techniques were applied to recommending text documents.
  - Like web pages or newsgroup messages for example.

- Content of items can also be represented as text documents.
  - With textual descriptions of their basic characteristics.
  - Structured: Each item is described by the same set of attributes

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Lace Reader</td>
<td>Fiction, Mystery</td>
<td>Brunonia Barry</td>
<td>Hardcover</td>
<td>49.90</td>
<td>American contemporary fiction, detective, historical</td>
</tr>
<tr>
<td>Into the Fire</td>
<td>Romance, Suspense</td>
<td>Suzanne Brockmann</td>
<td>Hardcover</td>
<td>45.90</td>
<td>American fiction, murder, neo-Nazism</td>
</tr>
</tbody>
</table>

- Unstructured: free-text description.
Content representation and item similarities

- **Item representation**

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Lace Reader</td>
<td>Fiction, Mystery</td>
<td>Brunonia Barry</td>
<td>Hardcover</td>
<td>49.90</td>
<td>American contemporary fiction, detective, historical</td>
</tr>
<tr>
<td>Into the Fire</td>
<td>Romance, Suspense</td>
<td>Suzanne Brockmann</td>
<td>Hardcover</td>
<td>45.90</td>
<td>American fiction, murder, neo-Nazism</td>
</tr>
</tbody>
</table>

- **User profile**

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Fiction</td>
<td>Brunonia, Barry, Ken Follett</td>
<td>Paperback</td>
<td>25.65</td>
<td>Detective, murder, New York</td>
</tr>
</tbody>
</table>

- **Simple approach**
  - Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
  - Or use and combine multiple metrics

\[
2 \times \frac{|keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}
\]
Term-Frequency - Inverse Document Frequency \((TF – IDF)\)

- Simple keyword representation has its problems
  - in particular when automatically extracted as
    - not every word has similar importance
    - longer documents have a higher chance to have an overlap with the user profile

- Standard measure: TF-IDF
  - Encodes text documents in multi-dimensional Euclidian space
    - weighted term vector
  - TF: Measures, how often a term appears (density in a document)
    - assuming that important terms appear more often
    - normalization has to be done in order to take document length into account
  - IDF: Aims to reduce the weight of terms that appear in all documents
**TF-IDF II**

- Given a keyword $i$ and a document $j$

- $TF(i, j)$
  - term frequency of keyword $i$ in document $j$

- $IDF(i)$
  - inverse document frequency calculated as $IDF(i) = \log \frac{N}{n(i)}$
    - $N$: number of all recommendable documents
    - $n(i)$: number of documents from $N$ in which keyword $i$ appears

- $TF - IDF$
  - is calculated as: $TF-IDF(i, j) = TF(i, j) \times IDF(i)$
Example TF-IDF representation

- **Term frequency:**
  - Each document is a count vector in $\mathbb{N}^{|\mathcal{V}|}$

<table>
<thead>
<tr>
<th>Term</th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>157</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>4</td>
<td>157</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>232</td>
<td>227</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1.51</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>worser</td>
<td>1.37</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Vector $\mathbf{v}$ with dimension $|\mathcal{V}| = 7$

Example taken from http://informationretrieval.org
Example TF-IDF representation

- Combined TF-IDF weights
  - Each document is now represented by a real-valued vector of \( TF-IDF \) weights \( \in \mathbb{R}^{|\nu|} \)

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>157</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caesar</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cleopatra</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mercy</td>
<td>1.57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>worser</td>
<td>1.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>5.25</td>
<td>3.18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.35</td>
</tr>
<tr>
<td>Brutus</td>
<td>1.21</td>
<td>6.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>8.59</td>
<td>2.54</td>
<td>0</td>
<td>1.51</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1.54</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>2.85</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1.51</td>
<td>0</td>
<td>1.9</td>
<td>0.12</td>
<td>5.25</td>
<td>0.88</td>
</tr>
<tr>
<td>worser</td>
<td>1.37</td>
<td>0</td>
<td>0.11</td>
<td>4.15</td>
<td>0.25</td>
<td>1.95</td>
</tr>
</tbody>
</table>

Example taken from http://informationretrieval.org
Improving the vector space model

- Vectors are usually long and sparse
- remove stop words
  - They will appear in nearly all documents.
  - e.g. "a", "the", "on", ...
- use stemming
  - Aims to replace variants of words by their common stem
  - e.g. "went" ⟷ "go", "stemming" ⟷ "stem", ...
- size cut-offs
  - only use top n most representative words to remove "noise" from data
  - e.g. use top 100 words
Improving the vector space model II

- Use lexical knowledge, use more elaborate methods for feature selection
  - Remove words that are not relevant in the domain

- Detection of phrases as terms
  - More descriptive for a text than single words
  - e.g. "United Nations"

- Limitations
  - Semantic meaning remains unknown
  - Example: usage of a word in a negative context
    - "there is nothing on the menu that a vegetarian would like.."
    - The word "vegetarian" will receive a higher weight than desired
    - An unintended match with a user interested in vegetarian restaurants
Cosine similarity

- **Usual similarity metric to compare vectors: Cosine similarity (angle)**
  - Cosine similarity is calculated based on the angle between the vectors
    \[ \text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|} \]

- **Adjusted cosine similarity**
  - take average user ratings into account \((\bar{r}_u)\), transform the original ratings
  - \(U\): set of users who have rated both items \(a\) and \(b\)
  \[ \text{sim}(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{ua} - \bar{r}_u)(r_{ub} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{ua} - \bar{r}_u)^2} \cdot \sqrt{\sum_{u \in U} (r_{ub} - \bar{r}_u)^2}} \]
Recommending items

- **Simple method: nearest neighbors**
  - Given a set of documents $D$ already rated by the user (like/dislike)
    - Either explicitly via user interface
    - Or implicitly by monitoring user's behavior
  - Find the $n$ nearest neighbors of an not-yet-seen item $i$ in $D$
    - Use similarity measures (like cosine similarity) to capture similarity of two documents
  - Take these neighbors to predict a rating for $i$
    - e.g. $k = 5$ most similar items to $i$.  
      4 of $k$ items were liked by current user $\implies$ item $i$ will also be liked by this user

- Variations:
  - Varying neighborhood size $k$
  - lower/upper similarity thresholds to prevent system from recommending items the user already has seen
- Good to model short-term interests / follow-up stories
- Used in combination with method to model long-term preferences
Recommending items

- Retrieval quality depends on individual capability to formulate queries with right keywords.

- **Query-based retrieval: Rocchio's method**
  - The SMART System: Users are allowed to rate (relevant/irrelevant) retrieved documents (feedback)
  - The system then learns a prototype of relevant/irrelevant documents
  - Queries are then automatically extended with additional terms/weight of relevant documents
Rocchio details

- **Document collections** $D^+$ (liked) and $D^-$ (disliked)
  - Calculate prototype vector for these categories.

- **Computing modified query** $Q_{i+1}$ from current query $Q_i$ with:

$$Q_{i+1} = \alpha \cdot Q_i + \beta \left( \frac{1}{|D^+|} \sum_{d^+ \in D^+} d^+ \right) - \gamma \left( \frac{1}{|D^-|} \sum_{d^- \in D^-} d^- \right)$$

- **$\alpha$, $\beta$, $\gamma$ used to fine-tune the feedback**
  - $\alpha$ weight for original query
  - $\beta$ weight for positive feedback
  - $\gamma$ weight for negative feedback

- **Often only positive feedback is used**
  - More valuable than negative feedback
Practical challenges of Rocchio's method

- Certain number of item ratings needed to build reasonable user model
  - Can be automated by trying to capture user ratings implicitly (click on document)
  - Pseudorelevance Feedback: Assume that the first $n$ documents match the query best. The set $D^{-}$ is not used until explicit negative feedback exists.

- User interaction required during retrieval phase
  - Interactive query refinement opens new opportunities for gathering information and
  - Helps user to learn which vocabulary should be used to receive the information he needs
Probabilistic methods

- **Recommendation as classical text classification problem**
  - long history of using probabilistic methods

- **Simple approach:**
  - 2 classes: hot/cold
  - simple Boolean document representation
  - calculate probability that document is hot/cold based on Bayes theorem

<table>
<thead>
<tr>
<th>Doc-ID</th>
<th>recommender</th>
<th>intelligent</th>
<th>learning</th>
<th>school</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
P(X|\text{Label} = 1) = P(\text{recommend} = 1|\text{Label} = 1) \
\times P(\text{intelligent} = 1|\text{Label} = 1) \
\times P(\text{learning} = 0|\text{Label} = 1) \
\times P(\text{school} = 0|\text{Label} = 1) \
= \frac{3}{3} \times \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3} \approx 0.149
\]
Linear classifiers

- Most learning methods aim to find coefficients of a linear model
- A simplified classifier with only two dimensions can be represented by a line

- The line has the form $w_1 x_1 + w_2 x_2 = b$
  - $x_1$ and $x_2$ correspond to the vector representation of a document (using e.g. TF-IDF weights)
  - $w_1$, $w_2$ and $b$ are parameters to be learned
  - Classification of a document based on checking $w_1 x_1 + w_2 x_2 > b$

- In n-dimensional space the classification function is $w^T \vec{x} = b$

- Other linear classifiers:
  - Naive Bayes classifier, Rocchio method, Windrow-Hoff algorithm, Support vector machines
Improvements

- **Side note: Conditional independence of events does in fact not hold**
  - "New York", "Hong Kong"
  - Still, good accuracy can be achieved

- **Boolean representation simplistic**
  - positional independence assumed
  - keyword counts lost

- **More elaborate probabilistic methods**
  - e.g., estimate probability of term v occurring in a document of class C by relative frequency of v in all documents of the class

- **Other linear classification algorithms (machine learning) can be used**
  - Support Vector Machines, ..

- **Use other information retrieval methods (used by search engines..)**
Explicit decision models

- **Decision tree for recommendation problems**
  - inner nodes labeled with item features (keywords)
  - used to partition the test examples
    - existence or non existence of a keyword
  - in basic setting only two classes appear at leaf nodes
    - interesting or not interesting
  - decision tree can automatically be constructed from training data
  - works best with small number of features
  - use meta features like author name, genre, ... instead of TF-IDF representation.
Explicit decision models II

- **Rule induction**
  - built on RIPPER algorithm
  - good performance compared with other classification methods
    - **elaborate postpruning techniques of RIPPER**
    - **extension for e-mail classification**
      - takes document structure into account

- **Main advantages of these decision models:**
  - inferred decision rules serve as basis for generating explanations for recommendation
  - existing domain knowledge can be incorporated in models
On feature selection

- process of choosing a subset of available terms

- different strategies exist for deciding which features to use
  - feature selection based on domain knowledge and lexical information from WordNet (Pazzani and Billsus 1997)
  - frequency-based feature selection to remove words appearing "too rare" or "too often" (Chakrabarti 2002)

- Not appropriate for larger text corpora
  - Better to
    - evaluate value of individual features (keywords) independently and
    - construct a ranked list of "good" keywords.

- Typical measure for determining utility of keywords: e.g. $X^2$, mutual information measure or Fisher's discrimination index
Limitations of content-based recommendation methods

- Keywords alone may not be sufficient to judge quality/relevance of a document or web page
  - up-to-date-ness, usability, aesthetics, writing style
  - content may also be limited / too short
  - content may not be automatically extractable (multimedia)

- Ramp-up phase required
  - Some training data is still required
  - Web 2.0: Use other sources to learn the user preferences

- Overspecialization
  - Algorithms tend to propose "more of the same"
  - Or: too similar news items
Discussion & summary

- In contrast to collaborative approaches, content-based techniques do not require user community in order to work

- Presented approaches aim to learn a model of user's interest preferences based on explicit or implicit feedback
  - Deriving implicit feedback from user behavior can be problematic

- Evaluations show that a good recommendation accuracy can be achieved with help of machine learning techniques
  - These techniques do not require a user community

- Danger exists that recommendation lists contain too many similar items
  - All learning techniques require a certain amount of training data
  - Some learning methods tend to overfit the training data

- Pure content-based systems are rarely found in commercial environments
Literature
