Content-based Filtering

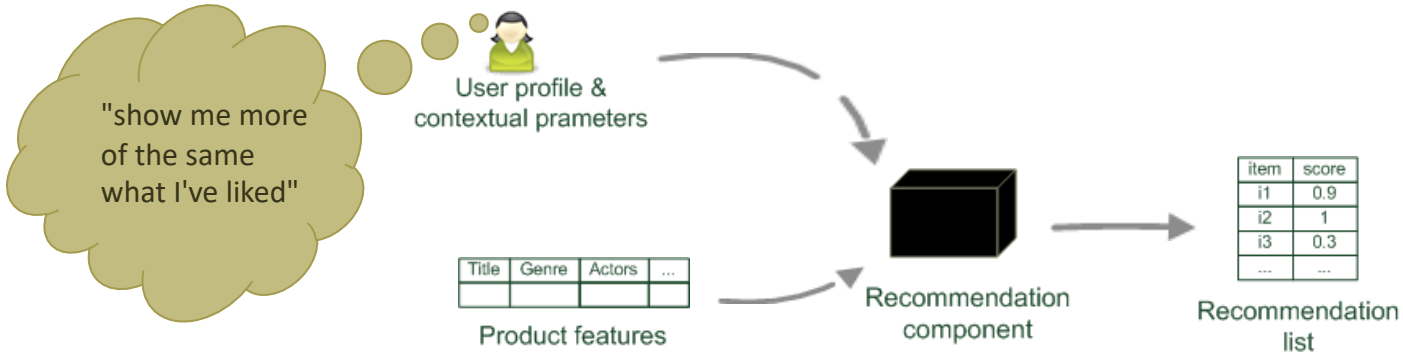
MEI

CONSTANTINO MARTINS, CATARINA FIGUEIREDO, DULCE MOTA AND JOAQUIM SANTOS

Disclaimer

- **Some of this material/slides are adapted from several:**
  - Presentations found on the internet;
  - Papers
  - Books;
  - Web sites
  - …

Content-Based Filtering



Content-Based Filtering

- Recommendations are based on information on the content of items rather than on other users' opinions
- Find similarity between items, in order to generate recommendations
- Vector space models like TF-IDF
- Probabilistic models such as Naive Bayes Classifier, decision trees, or neural networks can be used
- Machine learning algorithms can be employed to infer users' preferences from examples, based on a feature-rich description of content

Common Formal Model

X = set of Users

S = set of Items

Utility function u: X × S → R

- ✓ **R** = set of ratings
- ✓ **R** is a totally ordered set
- ✓ e.g., **0-5** stars, real number in **[0,1], ……**

Content-Based Filtering

- The Most CB-recommendation methods originate from Information Retrieval (IR) field (see the Course of the Master's Degree in Computer Engineering - ISEP):
  - ✓The item descriptions are usually automatically extracted (“important” words)
  - ✓Goal is to find and rank interesting text items/documents (news articles, web pages)
- Other considerations:
  - ✓Classical IR-based methods based on keywords
  - ✓No expert recommendation knowledge involved
  - ✓User profile (preferences) are rather learned than explicitly elicited

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

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

  - Unstructured: free-text description.

Simple approach - Dice coefficient

- Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
- Keyword Set: Each item (bi and bj) is associated with a set of keywords that describe its content, for example, keywords extracted from the item's title, description, or tags.

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, Murder, Neo-nazism
...					

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follet, ..	Paperback	25.65	detective, murder, New York

$$sim(b_i, b_j) = \frac{2 * |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

Dice coefficient

$$\text{sim}(b_i, b_j) = \frac{2 * |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

- Calculating Keyword Overlap: similarity metric that measures the overlap between two sets
- The formula is:
  - ✓ keywords(bi) represents the set of keywords associated with item bi
  - ✓ keywords(bj) represents the set of keywords associated with item bj
  - ✓ |keywords(bi) ∩ keywords(bj)| represents the size of the intersection between the keyword sets
  - ✓ |keywords(bi)| and |keywords(bj)| represent the sizes of the keyword sets, respectively
- The higher the value of the similarity calculated using the Dice coefficient, the greater the keyword overlap between items bi and bj. This indicates greater similarity in the description or content of the items.
- This simple approach is useful for recommending items based on keyword similarity, but it may not capture all relevant aspects of user preference

Example

- **Item representation**

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

- **User profile**

Title	Genre	Author	Type	Price	Keywords
…	Fiction	Brunonia, Barry, Ken Follett	Paperback	25.65	Detective, murder, New York

$keywords(b_j)$ describes Book b_j with a set of keywords

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

$$\frac{2 \times |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

Term-Frequency - Inverse Document Frequency (TF-IDF)

- It is a subfield of Natural Language Processing (NLP)
- Fundamentally, the occurrence of each word in a document is counted, the importance of each word is weighted, and the classification/score for the document is calculated
- The higher the TF-IDF score (weight), the rarer the term, and vice versa
- TF - Relative to the total number of words in the document, this is the occurrence of words in a document and the assignment of greater weight to higher frequency. This is divided by the size of the document (in number of words)

Term-Frequency - Inverse Document Frequency (TF-IDF)

- IDF (Inverse Document Frequency) is the ratio between the total number of documents and the frequency of occurrence of documents containing the word in question
- This defines the rarity of words, since as the occurrence of the word in the document decreases, the IDF increases
- It assists in assigning higher scores to rare terms in documents

TF-IDF

- Given a keyword t and a document d

$TF(t, d)$

  - ✓Term Frequency of keyword t in document d

$$TF_{t,d} = \frac{Frequência\ de\ ocorrência\ do\ termo\ t\ no\ documento\ d}{Número\ total\ de\ termos\ no\ documento}$$

TF-IDF

- If the same term 't' appears several times in the same document 'd', it is necessary to dampen the effect of high-frequency terms. For this purpose, the Weighted Term Frequency (Wt,d) for example is calculated for each term using the equation

$$W_{t,d} = \begin{cases} 1 + log_{10} TF_{t,d}, se\ TF_{t,d} > 0 \\ 0, de\ outra\ forma \end{cases}$$

TF-IDF

- Variants:

weighting scheme	TF weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
log normalization	$1 + \log(f_{t,d})$

- ✓Binary: 0 or 1 when the keyword is absent or present in the text
- ✓Raw count: count of how many times the term is present in the document
- ✓Term frequency: normalized counting such as the sum of the counts of all terms is equal to one
- ✓Logarithmic normalization: apply the logarithm to those counts, introducing a logarithmic scale for the counters, which may help to solve the task better

TF-IDF

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

$TF - IDF$

- Is calculated as: $\boldsymbol{TF\text{-}IDF(t,d) = TF(t,d) * IDF(t)}$
- Or $\boldsymbol{TF\text{-}IDF(t,d) = W(t,d) * IDF(t)}$

Bag of Words

- Each item is represented by a TF-IDF vector, whose score is obtained by applying equation
  - ✓That is, after defining the TF-IDF score of each term, term vectors are created for each item
- After calculating the TF-IDF scores, to determine which items are closest, the vector space model can be used
- To calculate the similarity between items, any of the methods already defined can be applied to the TF-IDF vectors: cosine similarity; Pearson correlation or Euclidean distance

Example TF-IDF representation

Term frequency:

- Each document is a count vector in $\mathbb{N}^{|v|}$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	1.51	0	3	5	5	1
worser	1.37	0	1	1	1	0

Vector v with dimension $|v| = 7$

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 $TF\text{-}IDF$ weights $\in \mathbb{R}^{|v|}$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4					
Caesar	232					
Calpurnia	0					
Cleopatra	57					
mercy	1.51					
worser	1.37					

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

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

- 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 then 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

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

- 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

$$sim(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$

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 ⇨ 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

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

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	1
2	0	0	1	1	0
3	1	1	0	0	1
4	1	0	1	1	1
5	0	0	0	1	0
6	1	1	0	0	?

$$
\begin{aligned}
&P(X|Label = 1) \\
&= P(recommender = 1|Label = 1) \\
&\times P(intelligent = 1|Label = 1) \\
&\times P(learning = 0|Label = 1) \\
&\times P(school = 0|Label = 1) = {}^{3}/_{3} \times {}^{2}/_{3} \times {}^{1}/_{3} \times {}^{2}/_{3} \\
&\approx 0.149
\end{aligned}
$$

Linear classifiers

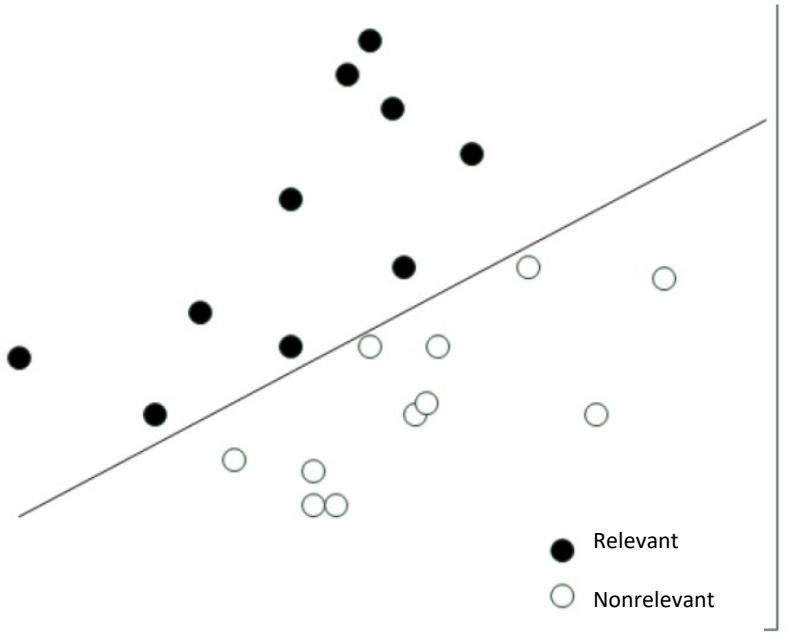
- Most learning methods aim to find coefficients of a linear model
- A simplified classifier with only two dimensions can be represented by a line

✓ **The line has the form $w_1x_1 + w_2x_2 = b$**

- 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

$$w_1x_1 + w_2x_2 > b$$

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



✓ **Other linear classifiers:**

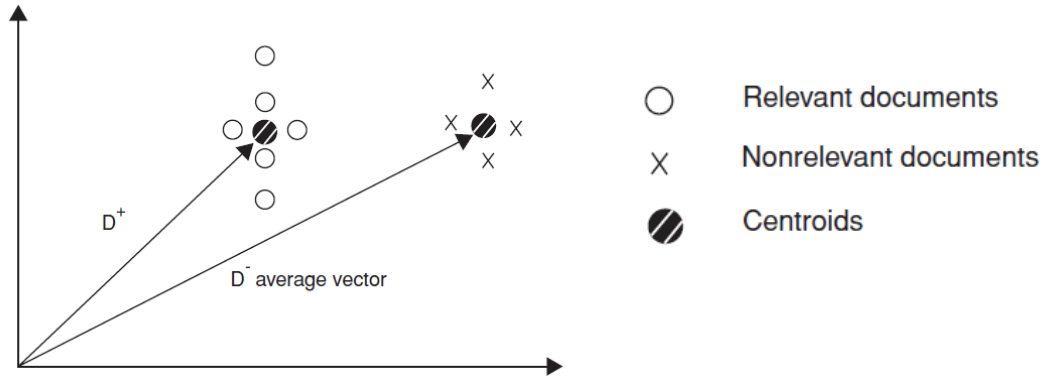
- Naive Bayes classifier, Rocchio method, Windrow-Hoff algorithm, Support vector machines

Recommending items

- Retrieval quality depends on individual capability to formulate queries with right keywords.
- Query-based retrieval: Rocchio's method
  - ✓Is a document classification algorithm that operates in vector space
  - ✓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:

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

- Often only positive feedback is used
  - More valuable than negative feedback

- α, β, γ used to fine-tune the feedback
  - α weight for original query
  - β weight for positive feedback
  - γ weight for 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

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.

Naïve Bayes

- Is derived from Bayes' theorem

- For Naïve Bayes, the naïve assumption is that:
  - ✓ Attributes/Characteristics/features are independent of each other, given the classes

Adapting Naïve Bayes idea for Book Recommendation

- Vector of Bags model
  - ✓ E.g. Books have several different fields that are all text
    - Authors, description, …
    - A word appearing in one field is different from the same word appearing in another
  - ✓ Want to keep each bag different—*vector of m Bags; Conditional probabilities for each word w.r.t each class and bag*

$$P(cj \mid Book) = \frac{P(cj)}{P(Book)} \prod_{m=1}^{S} \prod_{i=1}^{|dm|} P(a_{mi} \mid cj, sm)$$

- Can give a profile of a user in terms of words that are most predictive of what they like
  - ✓ Odds Ratio

    P(rel|example)/P(~rel|example)

    An example is positive if the odds ratio is > 1
  - ✓ Strengh of a keyword
    - Log[P(w|rel)/P(w|~rel)]
      - We can summarize a user's profile in terms of the words that have strength above some threshold.

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

Conclusion

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

References

Serão colocadas depois da entrega do primeiro trabalho