Content-Based Recommenders

Recommender Systems

Dimitris Sacharidis

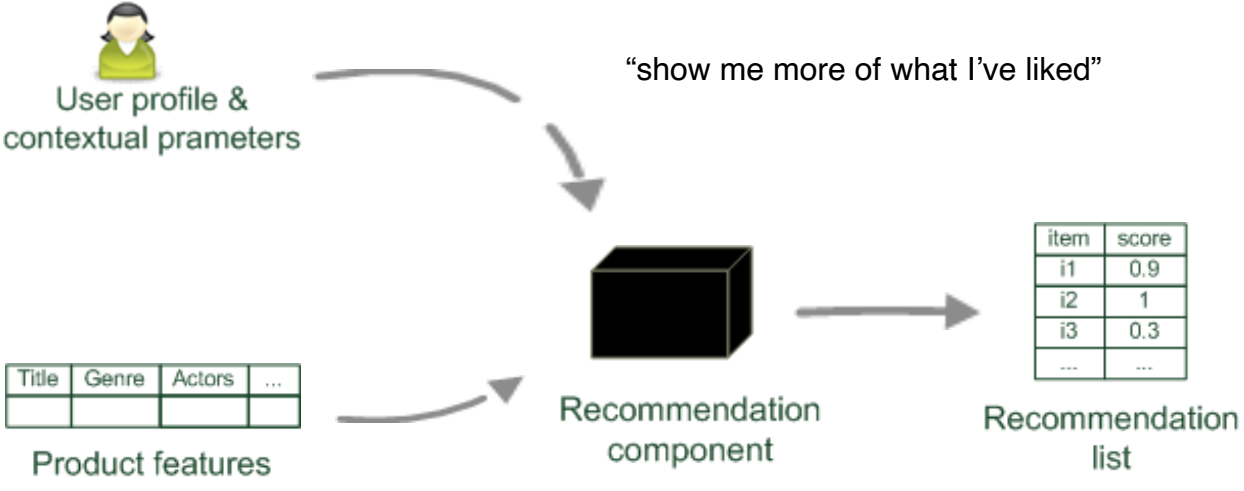
Acknowledgements

some content from:

- Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich. Recommender Systems: An Introduction
- Amra Delić

Content-Based Recommender Systems

- there is **no collaboration**, but there is feedback **history**
- the system recommends items **similar** to those the user has liked in the past
- similarity is based on the **content** of the items



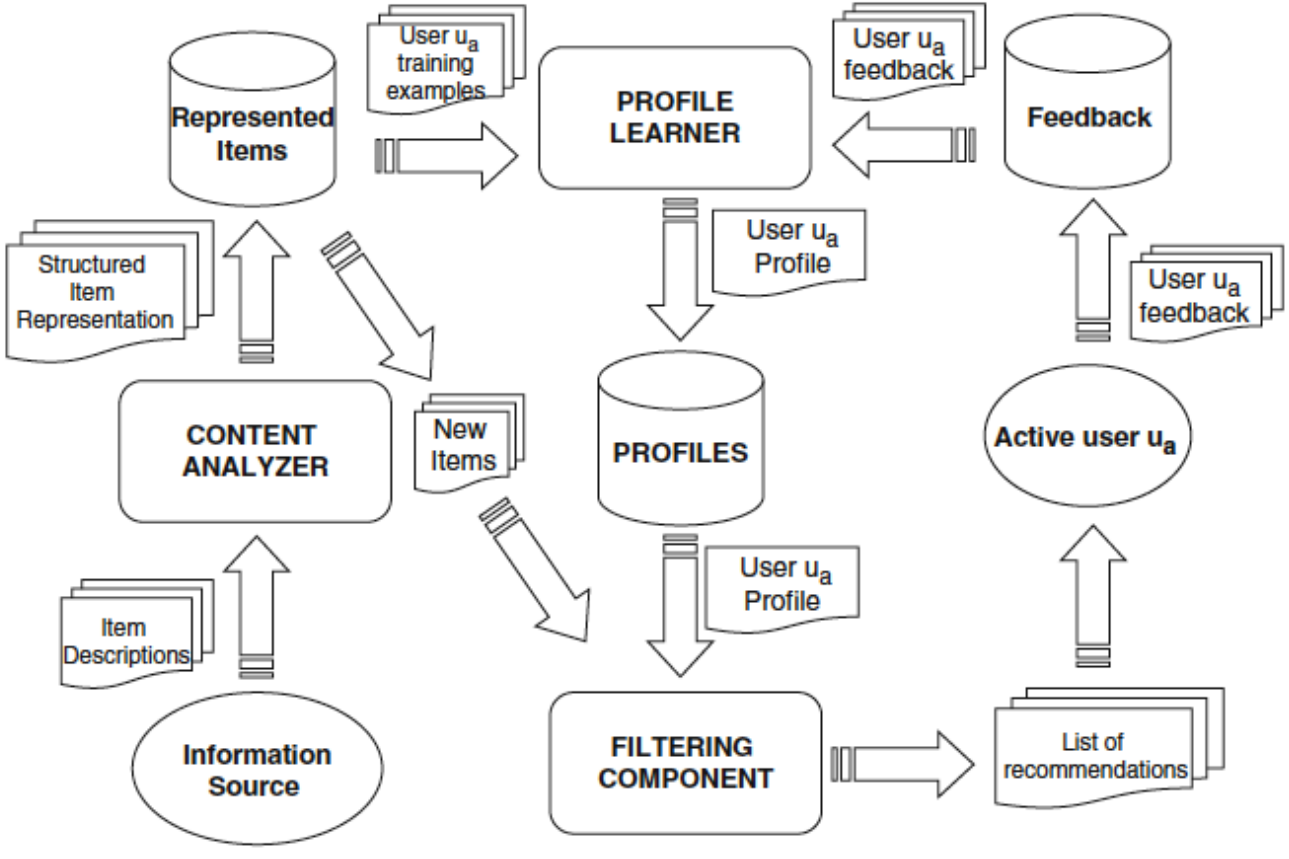
Terminology

- **Item content** is a description of the item in terms of some features
  - task of the **content analyzer**
  - different ways to define this

- **User profile** is a description of the user's preferences in terms of the item content in her feedback history
  - task of the **profile learner**
  - different ways to define this

- Content-based recommender estimates how much the user profile **matches** to an item content
  - task of the **filtering component**
  - different ways to achieve this

Workflow



[2015 F. Ricci et al.] *Recommender Systems Handbook, Chapter 3*

Item Content

What can be item content

- Items are usually described by the same **set of attributes**/features
  - attributes can be **structured**, e.g., price, type
  - or **non-structured**, e.g., genre, keywords, text

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

- but we need a more suitable representation for item content

vector representation of items

- goal is to **represent** each **item** by a **vector**
  - why? easier to handle, know how to define similarity

- for some attributes (*numerical* like price, *categorical* like type) conversion is easy

- for non-structured attributes (*sets* of keywords, genres; *text*) conversion uses *information retrieval* (IR) techniques
  - we will cover some basic ideas

vector space model

- goal is to convert a non-structured attribute (simply call it the document) into a vector

- **document** *d* is represented by a *n*-dimensional **vector**

$$\langle w_{1d}, w_{2d}, \ldots, w_{nd} \rangle$$

- where w_{td} is the **weight** for term *t* in document *d*
- and *n* is the number of **terms**

vector space model

- 3 simple steps to go from document to vector

1. assume **bag-of-words** semantics
2. count **frequencies** of words
3. **weigh**/normalize frequencies

- more modern approaches exist that provide **dense** representations (e.g., doc2vec)
  - they will not be covered here

bag of words

- ignore the ordering of words in a document
- “John is quicker than Mary” and ”Mary is quicker than John” become the same bag
- this may seem restrictive, but in practice it captures the “general meaning” of the document

count frequencies

- consider 6 documents (plays by Shakespeare)
- count the frequencies of 7 words

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	1
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	2	0	3	5	5	1
worser	2	0	1	1	1	0

a document represented as a frequency vector

weigh frequencies

- does the **frequency** of a word represent its **significance**?

- two problems with raw frequencies

1. Significance does not increase proportionally with frequency
   - a document with 10 occurrences of the word **is more relevant** than a document with 1 occurrence of the word. but **not 10 times** more relevant!

2. Significance depends on the overall frequency of a word in a collection of documents
   - articles and other stop words are not that relevant

weigh frequencies

- we will compute the weight of a term/word *t* in document *d* as the product of two terms:

$$w_{t,d} = tf_{t,d} * idf_t$$

- $tf_{t,d}$ is called **term frequency** (solves prob. 1)
- idf_t is called **inverse document frequency** (solves prob. 2)

- this is called the tf-idf scheme in information retrieval

term frequency

- Gustav Fechner (1801 - 1887) was obsessed with the relation of mind and matter
- Variations of a physical quantity (e.g. intensity of light) cause variations in the perceived subjective experience

- Fechner proposed that in many cases the connection is **logarithmic**
  - An increase of stimulus intensity by a given factor (say 10 times) always yields the same increment on the psychological scale
- If raising the frequency of a term from 10 to 100 increases relevance by 1 then raising the frequency from 100 to 1000 also increases relevance by 1

term frequency

- the **term frequency (tf)** of term *t* in document *d* is computed as the logarithm of the raw term frequency $f_{t,d}$

$$tf_{t,d} = \begin{cases} \log_{10}(1 + f_{t,d}), & if \quad f_{t,d} > 0 \\ 0, & otherwise \end{cases}$$

- examples for raw frequency → term frequency:
  - 0 → 0, 1 → 0.3, 10 → ~1, 1000 → ~3, etc.

inverse document frequency

- **rare terms** – in the whole collection – are **more informative** than frequent terms

- stop words carry almost no information
- but rare terms like *trypophobia* are usually very important

- we want to reward such infrequent terms

inverse document frequency

- the document frequency df_t of term *t* is the number of documents that contain *t*
  - the less frequent a term is, the lower its document frequency is

- we want to reward small values of document frequency

- the **inverse document frequency (idf)** is calculated as:

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

  - where *N* is the number of documents
  - idf does not depend on a document

inverse document frequency

- suppose N = 1.000.000 documents in the collection
- there is one idf value for each term in a collection

term	df_t	idf_t
Calpurnia	1	6
animal	100	4
Sunday	1.000	3
fly	10.000	2
under	100.000	1
the	1.000.000	0

tf-idf weighting

- term *t* in document *d* has a weight computed as:

$$w_{t,d} = tf_{t,d} * idf_t$$

- or equivalently:

$$w_{t,d} = \log_{10}(1 + f_{t,d}) * \log_{10}\left(\frac{N}{df_t}\right)$$

- the weight increases:
  - with the number of occurrences within a document
  - with the rarity of the term in the collection

final vectors

- from **raw frequency** vectors to **tf-idf** vectors
  - (assuming only these N=6 documents)

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	0.662	0.563	0	0	0	0.091
Brutus	0.210	0.662	0	0.091	0	0
Caesar	0.187	0.187	0	0.038	0.024	0.024
Calpurnia	0	0.810	0	0	0	0
Cleopatra	1.372	0	0	0	0	0
mercy	0.038	0	0.048	0.062	0.062	0.024
worser	0.084	0	0.053	0.053	0.053	0

User Profiles + Matching

User Profiles + Matching

- In general the problem is a **classification** task:
  - decide whether an **item content** matches the **user profile**, i.e., is the item **relevant** for the user?

- We will discuss two ways to construct user profiles, and appropriate matching methods
  - k-nearest neighbors
  - relevance feedback

User Profiles + Matching

k-nearest neighbors

k-nearest neighbors

- very similar to item-item collaborative filtering
- main difference is on the item **similarity**
  - CF uses **rating vectors**
  - CB uses **item content**

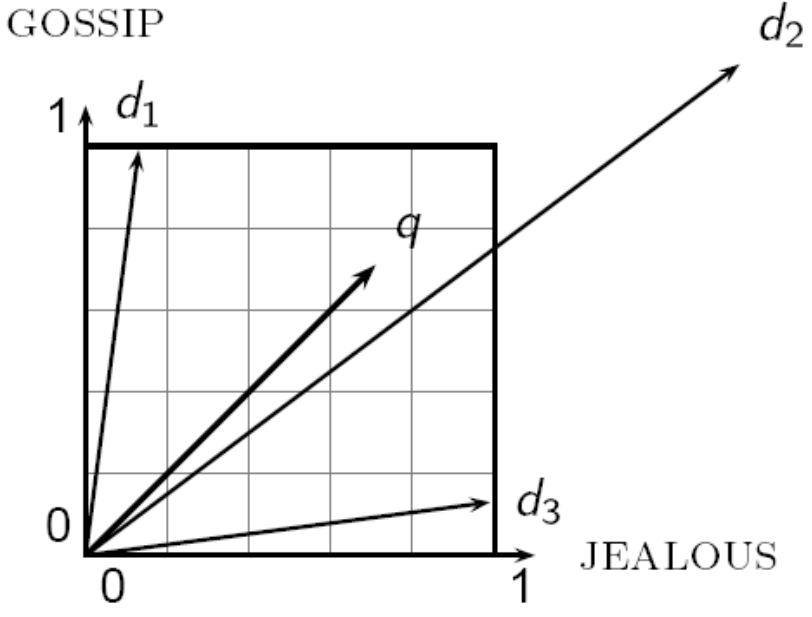
- **User profile**:
  - the set of vectors for all past rated items

- **Matching**:
  - the target item vector is compared to each past rated item vector
  - a similarity score is computed
  - compute the similarity-weighted average of the rating

similarity in the vector space model

- small Euclidean distance is a bad idea
  - why?

- which item is most similar to q?
- Euclidean says d_1
- but we probably want d_2
  - why? distribution of terms is more similar



similarity in the vector space model

- thought experiment: take an item *d* and append it to itself, thus create a new item *d′*
- "semantically" *d* and *d′* have the same content
- **Euclidean distance** between the two items can be quite large
- but the **angle** between the two items is zero
- what if we normalize vectors so that all lie in a (hyper-) sphere?
  - would Euclidean work then?

similarity in the vector space model

- **cosine similarity** is the cosine of the **angle** between two vectors, say past rated item d and target item q
- can be computed easily:

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\cdot\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

  - q_i is the tf-idf weight of term i in the target item
  - d_i is the tf-idf weight of term i in the past rated item

- if vectors are normalized, cosine similarity is simply the inner product of the vectors

similarity-weighted average

- for the target item q, determine its *k* **neighbors** among the past rated items in the user profile
- compute the **similarity-weighted average** rating in the neighborhood
  - looks familiar?

$$\hat{r}_{uq} = \frac{\sum_{d \in N(q;u)} \cos(q,d) \cdot r_{ud}}{\sum_{d \in N(q;u)} \cos(q,d)}$$

User Profiles + Matching

relevance feedback

relevance feedback

- users **feed back** the system with the **relevance** of retrieved documents
  - adopted in Information Retrieval

Rocchio's method

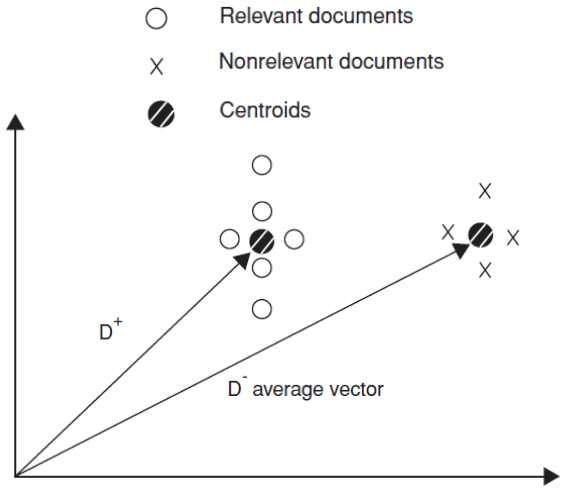
- **User profile**:
  - a *single* vector that is a linear combination of vector of past rated items

- **Matching**:
  - the similarity of the target item to the user profile is computed

relevance feedback

- past rated items are split into two classes:
  - D^+ positive feedback (relevant items)
  - D^- negative feedback (non relevant items)

- user profile vector:
- linear combination of three components:
  - some **baseline vector**
  - a **prototype for the positive class**
  - a **prototype for the negative class**



$$u = \alpha \cdot u_0 + \beta \cdot \frac{1}{|D^+|} \sum_{d^+ \in D^+} d^+ - \gamma \cdot \frac{1}{|D^-|} \sum_{d^- \in D^-} d^-$$

relevance feedback

matching is simple:

- compute the cosine similarity of the **user profile vector** to the **target item**
  - this is the expected degree of relevance to the user

discussion

benefits

- **user independence**: ratings from other users are not needed for the method to deliver recommendations

- **no cold-start problem for items**: relevance is determined by the item content; no feedback is required
  - works for dynamic items that are generated on the fly, or never before seen

- **transparency**: recommendations can be explained by listing the items that user previously liked

drawbacks

- **not for all domains**: you need good features; sometimes hard/impossible to get (e.g., movies, music)

- **overspecialization**: the user gets recommended only similar items to those already rated; “more of the same” (*filter bubble*)
  - no *novelty*; no *serendipity* (= being positively surprised by recommendations)

- **requires feedback**: some user feedback is necessary

Acknowledgements

some slides from:

- Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich. Recommender Systems: An Introduction