Master on Foundations of Data Science

Recommender Systems

Content Based Recommendations

Santi Seguí | 2019-2020

Today:

1) Do you have any question about task #2 or previous class?

2) Content-Based Recommendation

Content-Based Methods

Conceptual goal:

Give me recommendation based on the content (attributes)

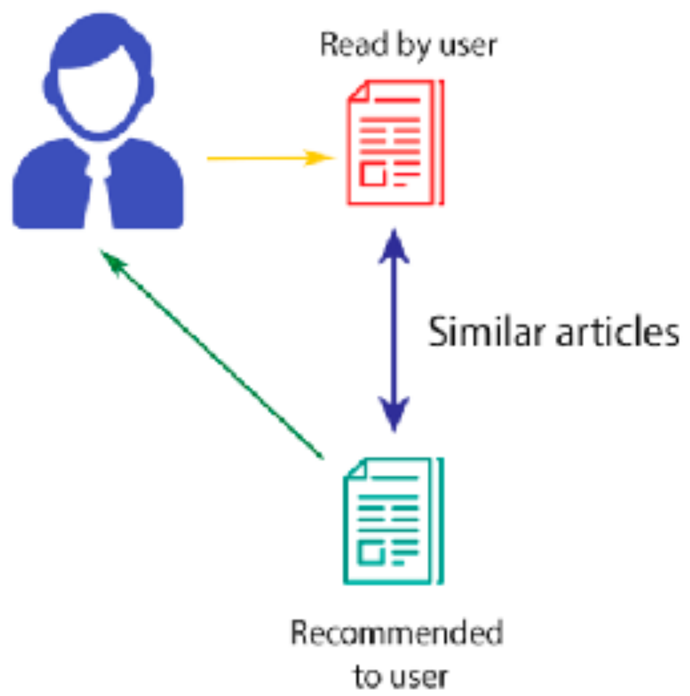
I liked before

Input:

User ratings (user profile)

+

item attributes (item profile)



When Content Based?

Really popular for **cold-start** problems.

Popular in domain like:
news recommendation or **music** recommendation

http://eigentaste.berkeley.edu/

First rate two jokes.

Q: If a person who speaks three languages is called "trilingual," and a person who speaks two languages is called "bilingual," what do you call a person who only speaks one language?

A: American!

Less Funny More Funny

Next

Advantages of CBRS

- **User independence**
  - CBRS exploit solely ratings provided by the active user to build the recommendation
  - No need for data on other users
- **Transparency**
  - Can provide explanations for recommended items by listing content-features that caused an item to be recommended
- **New Item (Cold Start on items)**
  - Can recommend new and unknown items

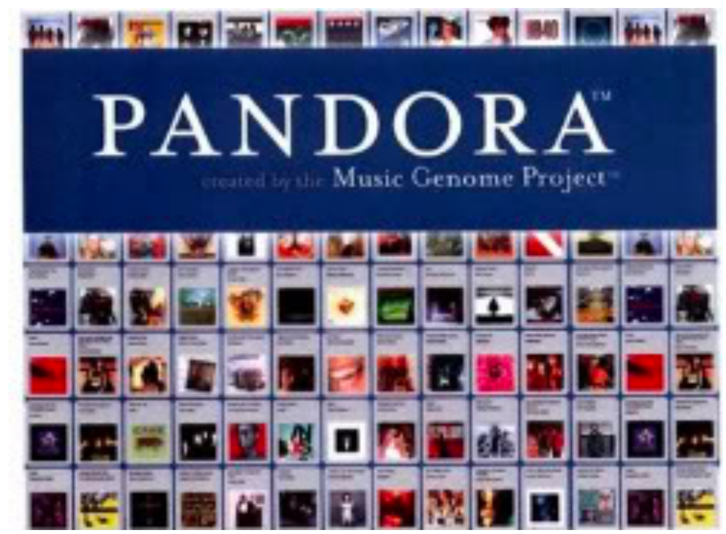
Some Famous CB Recommender Systems

https://www.pandora.com/

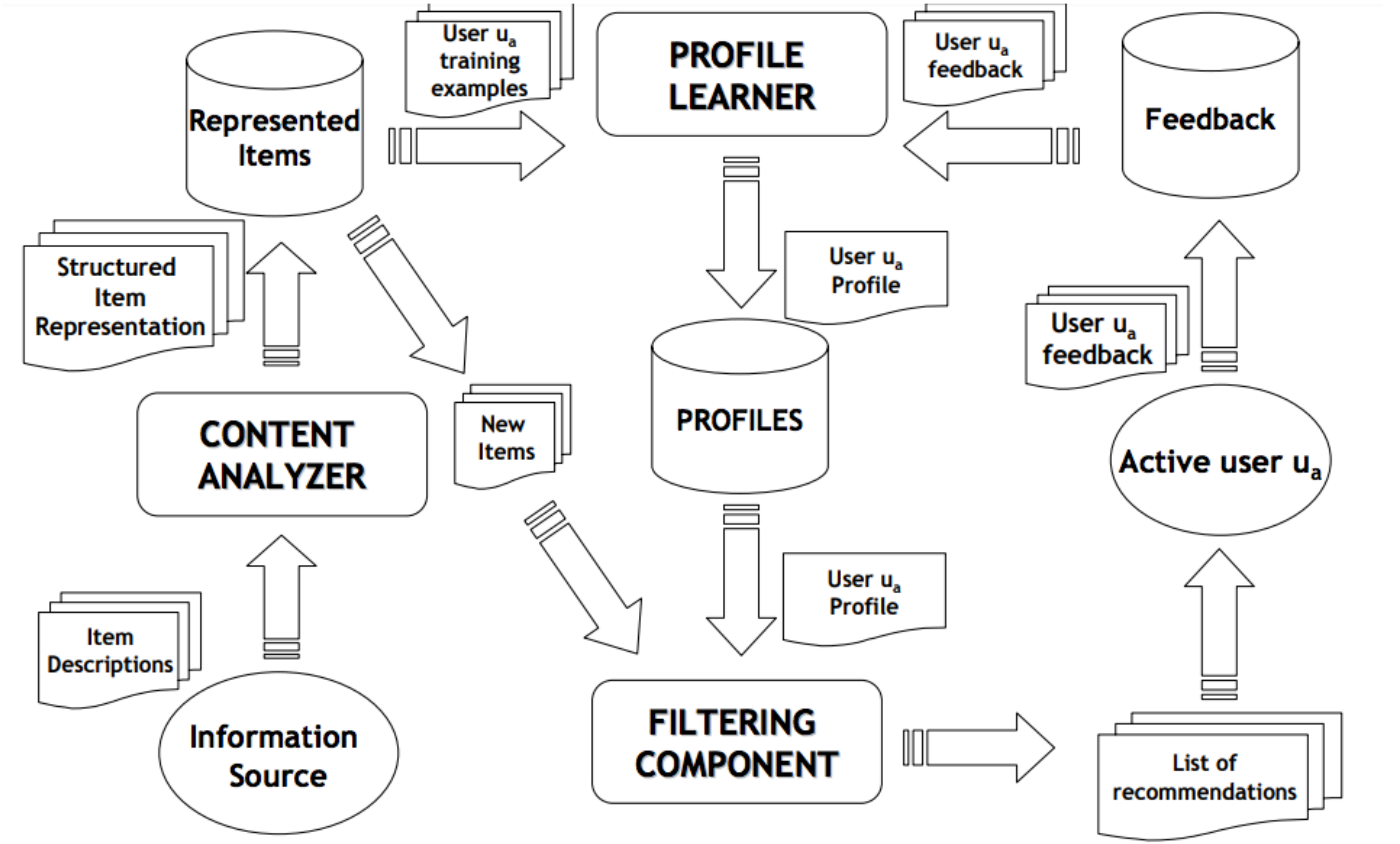
Pandora

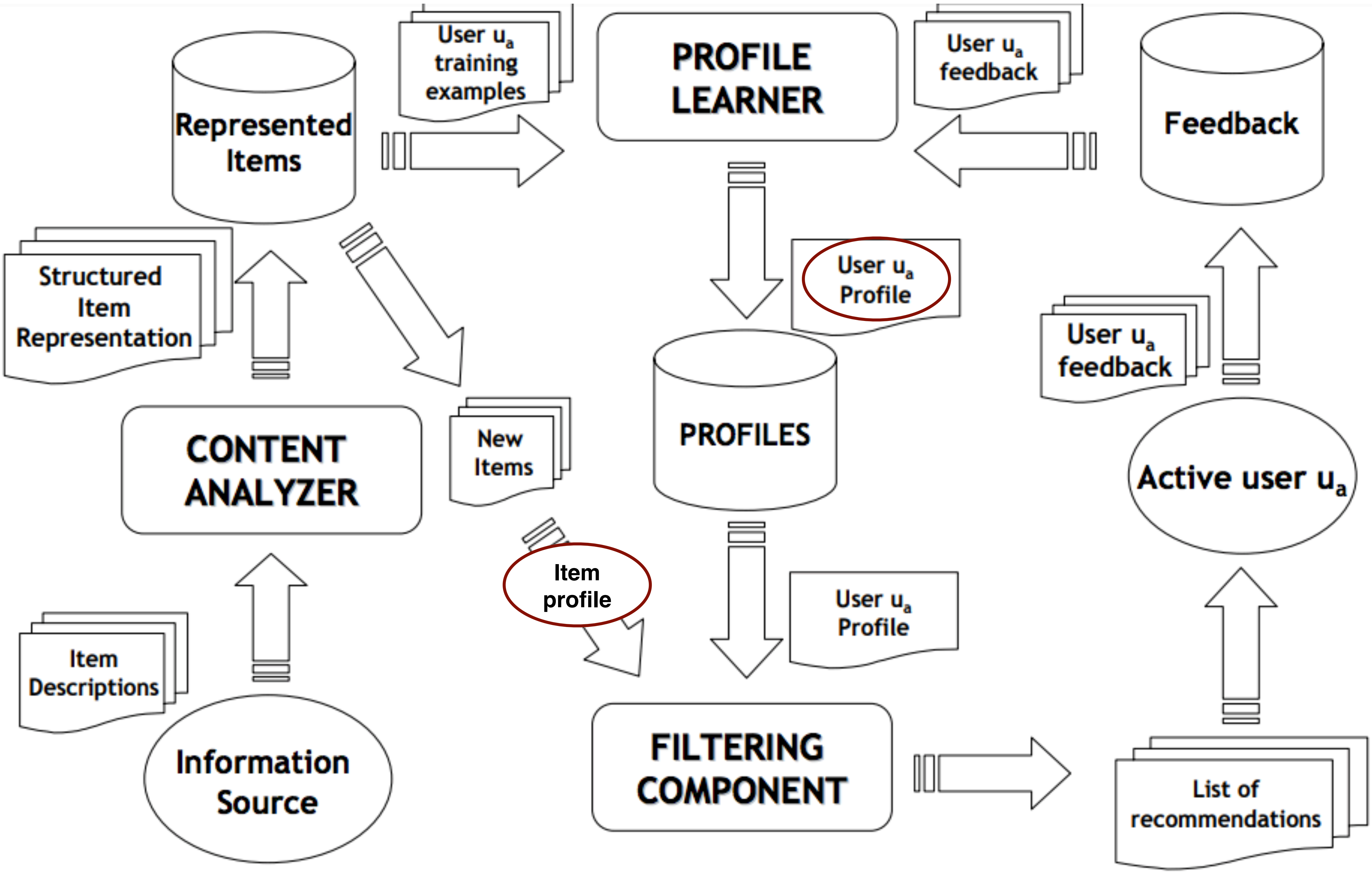
- How it works:
  - Base it recommendation on data

    from Music Genome Project



  - Assigns 400 attributs for each song, done by musicians.
    - Some reports say it takes half an hour per second of audio
  - Use this method to find sons which are silimar to the users's favorite sogs


Represented Items
User u_a training examples
PROFILE LEARNER
User u_a feedback
Feedback
Structured Item Representation
User u_a Profile
User u_a feedback
CONTENT ANALYZER
New Items
PROFILES
Active user u_a
User u_a Profile
Item Descriptions
Information Source
FILTERING COMPONENT
List of recommendations


User u_a training examples
PROFILE LEARNER
User u_a feedback
Represented Items
Feedback
Structured Item Representation
User u_a Profile
User u_a feedback
CONTENT ANALYZER
New Items
PROFILES
Active user u_a
Item profile
User u_a Profile
Item Descriptions
Information Source
FILTERING COMPONENT
List of recommendations

Key point:

Similar Items must have
similar profile/vector **representation**

to do so, we need rich features and smart ecoding

Item Profile

Item profile

- For each item, create an item profile
- Profile is a set of features.
    - Which features??
        - **Movies**: author, title, director, actor,…
        - **Images**, **videos**: raw content, metadata and tags
        - **People**: set of friends
        - **News**: keywords,..
- Convenient to consider the item profile as a vector:
    - One entry per feature (e.g., each actor, director, ..)
    - Vector might be boolean or real-valued


We want to create a **Hotel Recommendation** system based on content

We have to create the item profile

Which features should we use?

We want to create a **Hotel Recommendation** system based on content

from each hotel we have several features:

- City
- Location
- Price
- #Stars
- Swimming Pool?

How should be your feature vector?

And your user vector?

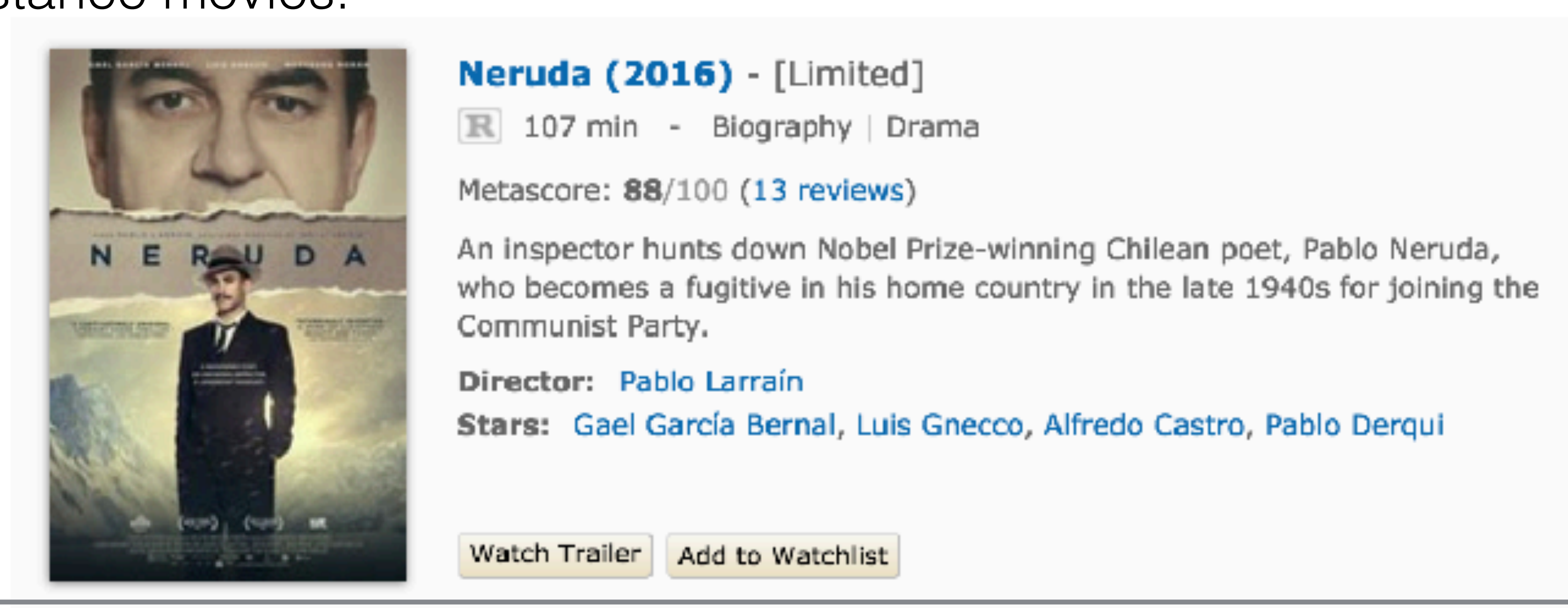
Content-based Filtering

- Requires **content** (from the items) that can be encoded as meaningful **features**.
  - Item title, description, price, image, etc…
- Need to compute a **similarity between items** based on the content of the items.
- **Users' tastes** must be represented as a **learnable function** of these content features.
- Does **not** to exploit quality judgments of **other users**.
  - Unless these are somehow included in the content features.

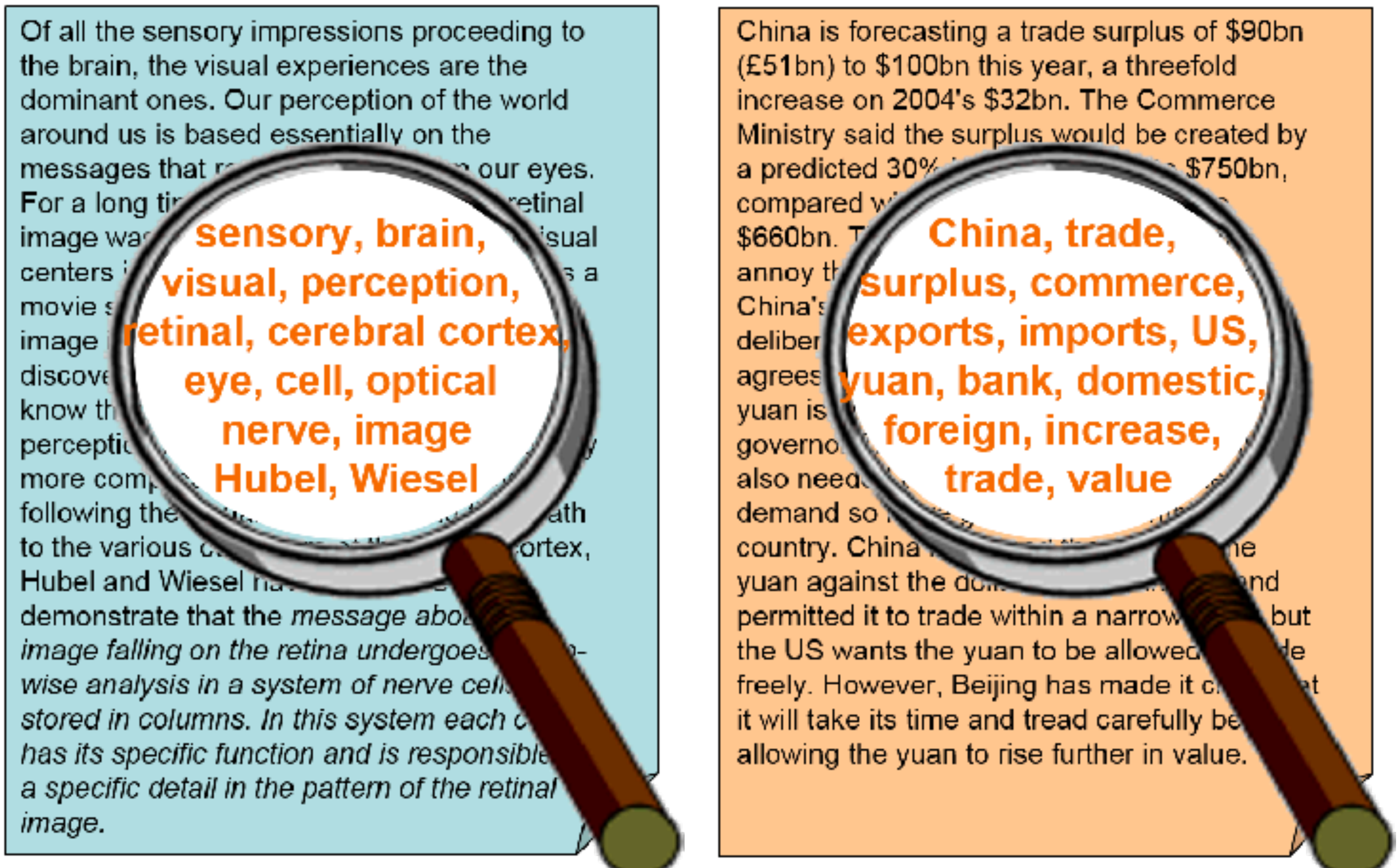
What is "content"?

- Content Based recommenders systems have been applied mosty on **text document**
- However, content of items, items such as movies or songs, can be represented as text documents
  - With textual description of their basics characteristics
  - Structured: Each item is described with the same set of attributes
  - Unstructured: Free-text document

As for instance movies:



How to describe textual information?



Feature Representation and Cleaning

- Extremly important when the unstructured format is used for representation.
- Bag of Words (BOW) from the unstructured description of the products or Web Pages used to be used, however, these reprensentrations needs to be cleaned and represented in a suitable format for processing.
- Several important steps:
  - **Stop-word removal**: Words such “a”, “an,”, “the”, does not provide important information
  - **Stemming**. Variations of the same words are consolidated. For example, words such “hope” and “hoping” are consolidated into the common root “hop”
  - **Phrase extraction:** The idea is to detect words that occur together in documents on a frequent basis.

TF-IDF

In information retrieval, **tf–idf**, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining.

- **Term Frequency** x **Inverse Document Frequency**
- **Term Frequency** =
  - Number of occurrences of a term in a document (can be a simple count)
- **Inverse Document Frequency** =
  - How few documents contain this term
  - Typically : Log(#documents / #documents with term)

So, items that appears rarely or appears everywhere are not important

TD-IDF

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

$$\mathrm{tf}(''\mathsf{this}'', d_1) = \frac{1}{5} = 0.2 \qquad \mathrm{tf}(''\mathsf{this}'', d_2) = \frac{1}{7} \approx 0.14$$

idf is constant per corpus, and **accounts** for the ratio of documents that include the word "this". In this case, we have a corpus of two documents and all of them include the word "this".

$$\mathrm{idf}(''\mathsf{this}'', D) = \log\left(\frac{2}{2}\right) = 0$$

tf–idf is zero for the word "this", which implies that the word is not very informativ as it appears in all documents.

$$\mathrm{tfidf}(''\mathsf{this}'', d_1) = 0.2 \times 0 = 0 \qquad \mathrm{tfidf}(''\mathsf{this}'', d_2) = 0.14 \times 0 = 0$$

What does TFIDF do?

- Automatic find of stop words, common terms
- Promote core terms over accidental terms
- When it fails?
  - If core term is not used frequently in a document (e.g., legal contracts)

Variants and Alternatives

- Some applications use variants on TF
  - Binary
  - Logarithmic frequencies
  - Normalized frequencies (log(tf + 1))

Relevance and Problems

- Significance in Documents
  - Titles, heading,… (different weight?)
- Phrases and n-grams
  - “recommender system” != “recommender” and “system”
  - Adjacency
- General score
- Implied Contet
  - Links, usage,…

Keyword Vector

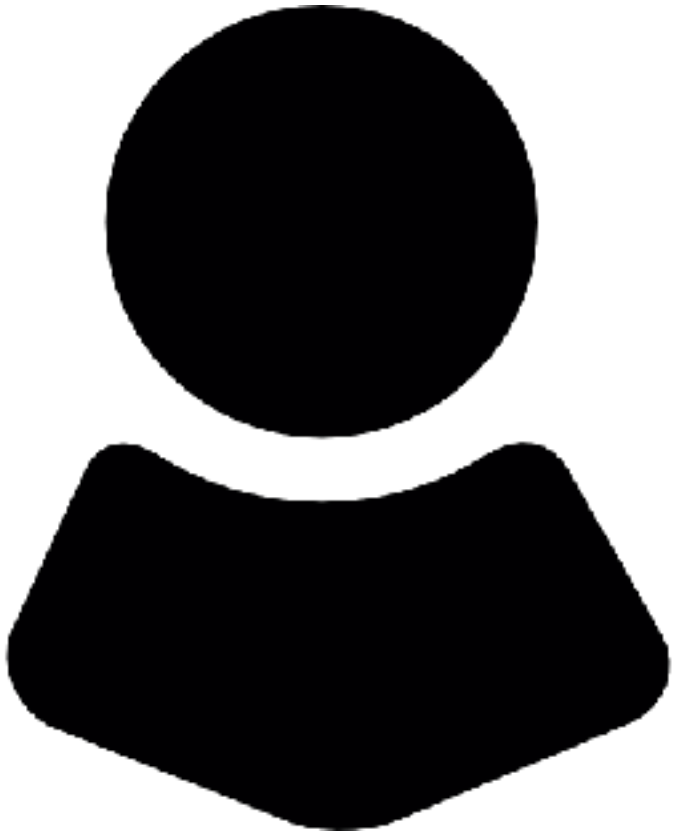
- The universe of possible keywords defines a content space
  - Each “keyword” is a dimension
  - Each item has a position in that space; that position defines a vector
  - Each user has a taste profile that is also a vector in that space
  - The match between user preferences and items is measured by how closely the two vectors aling
  - May want to limit/collapse keyword space

Vector representation

- Simple 0/1 (keyword applies or does not)
- Simple occurrence count
- TF-IDF
- Other variants include factors such as document length
- Eventually, this vector is ofter normalized

Other terms?

- Clothing attributes (color, size, etc..)
- Terms used in hotel reviews (pool, front desk, child friendly)
- Terms used in news articles (elections, football, economy)



User Profile

User Preferences

- My preferences:
    - **Movies** - I like SeVeN, American History X, Gladiator
    - **Hotels** - I prefer 24-hour front desk, internet, spa
    - **Music** - I like Blur, Pulp, The Verve,…

User Preferences

- **User Vector Space Model:**
  - single scalar value for each dimension
  - same dimensions than **item Vector Space**

How to build preferences?

- count the number of times the user chooses item with each keyword
- Set of “keyword” a user  or 
- or more sophisticated methods

UNIVERSITAT DE BARCELONA

User Preferences

- How to **accumulate** features from the profiles?
  - Add together the item vectors?
    - Should we normalize first?
      - Should all items have the same weight?
    - Should we weight the vectors somehow?
      - We can use ratings..
      - Confidence?

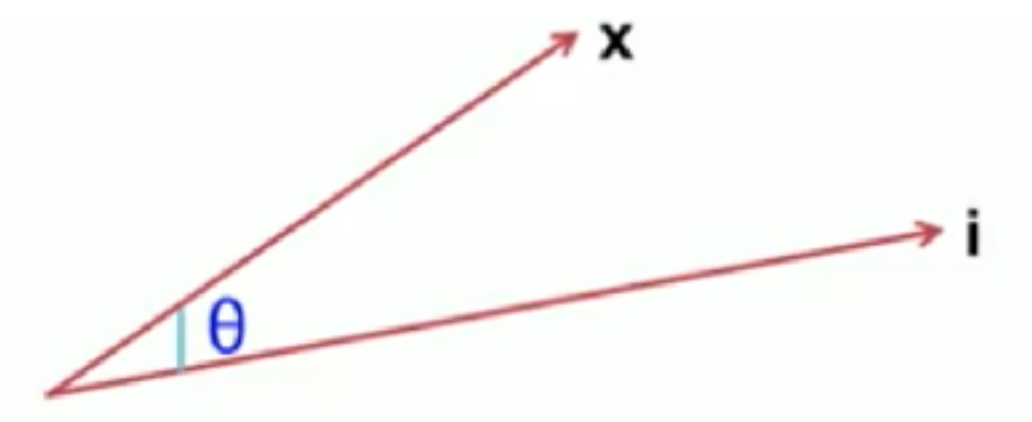
User Preferences

- What about **new user feedback from items?**
  - Update the user vector taking into account number of items used
  - Depending on the problem update with some decay

Making predictions

Making Predictions

- User profile x, Item profile i
- Estimate U(x,i) = cos(ø) = (x . i) / (|x| |i|)



- Technically, the cosine distance is actually the angle ø. And the cosine similarity is the angle 180-ø

How to improve it?

- Better classifier/Regressor
  - Liner regression to XGboost models are used
  - Each feature will have a different weight on the recommendation
- **Richer representations**

ACM RecSys Challenge 2020

ACM RecSys Challenge 2020

About Participation Timeline Organizers RecSys 2020

Dataset description

The Data is available to download here. Fields in each data entry are separated by the 1 character (*0x31 in UTF-8*) and each data entry will be characterized by the following features:

	Feature Name	Feature Type	Feature Description
Tweet Features	Text tokens	*List[long]*	Ordered list of Bert ids corresponding to Bert tokeniza
	Hashtags	*List[string]*	Tab separated list of hastags (identifiers) present in th
	Tweet id	*String*	Tweet identifier
	Present media	*List[String]*	Tab separated list of media types. Media type can be i
	Present links	*List[string]*	Tab separated list of links (identifiers) included in the
	Present domains	*List[string]*	Tab separated list of domains included in the Tweet (t
	Tweet type	*String*	Tweet type, can be either Retweet, Quote, Reply, or To
	Language	*String*	Identifier corresponding to the inferred language of th
	Timestamp	*Long*	Unix timestamp, in sec of the creation time of the Twe
Engaged With User Features	User id	*String*	User identifier
	Follower count	*Long*	Number of followers of the user
	Following count	*Long*	Number of accounts the user is following
	Is verified?	*Bool*	Is the account verified?
	Account creation time	*Long*	Unix timestamp, in seconds, of the creation time of th
Engaging User Features	User id	*String*	User identifier
	Follower count	*Long*	Number of followers of the user
	Following count	*Long*	Number of accounts the user is following
	Is verified?	*Bool*	Is the account verified?
	Account creation time	*Long*	Unix timestamp, in seconds, of the creation time of th
Engagement Features	Engagee follows engager?	*Bool*	Does the account of the engaged tweet author follow
	Reply engagement timestamp	*Long*	If there is at least one, unix timestamp, in s, of one of t
	Retweet engagement timestamp	*Long*	If there is one, unix timestamp, in s, of the retweet of t
	Retweet with comment engagement timestamp	*Long*	If there is at least one, unix timestamp, in s, of one of t
	Like engagement timestamp	*Long*	If there is one, Unix timestamp, in s, of the like

How can we create a tweet similarity?

ACM RecSys Challenge 2020

About Participation Timeline Organizers RecSys 2020

Dataset description

The Data is available to download here. Fields in each data entry are separated by the 1 character (*0x31 in UTF-8*) and each data entry will be characterized by the following features:

	Feature Name	Feature Type	Feature Description
Tweet Features	Text tokens	*List[long]*	Ordered list of Bert ids corresponding to Bert tokeniza
	Hashtags	*List[string]*	Tab separated list of hastags (identifiers) present in th
	Tweet id	*String*	Tweet identifier
	Present media	*List[String]*	Tab separated list of media types. Media type can be i
	Present links	*List[string]*	Tab separated list of links (identifiers) included in the
	Present domains	*List[string]*	Tab separated list of domains included in the Tweet (t
	Tweet type	*String*	Tweet type, can be either Retweet, Quote, Reply, or To
	Language	*String*	Identifier corresponding to the inferred language of th
	Timestamp	*Long*	Unix timestamp, in sec of the creation time of the Twe
Engaged With User Features	User id	*String*	User identifier
	Follower count	*Long*	Number of followers of the user
	Following count	*Long*	Number of accounts the user is following
	Is verified?	*Bool*	Is the account verified?
	Account creation time	*Long*	Unix timestamp, in seconds, of the creation time of th
Engaging User Features	User id	*String*	User identifier
	Follower count	*Long*	Number of followers of the user
	Following count	*Long*	Number of accounts the user is following
	Is verified?	*Bool*	Is the account verified?
	Account creation time	*Long*	Unix timestamp, in seconds, of the creation time of th
Engagement Features	Engagee follows engager?	*Bool*	Does the account of the engaged tweet author follow
	Reply engagement timestamp	*Long*	If there is at least one, unix timestamp, in s, of one of t
	Retweet engagement timestamp	*Long*	If there is one, unix timestamp, in s, of the retweet of t
	Retweet with comment engagement timestamp	*Long*	If there is at least one, unix timestamp, in s, of one of t
	Like engagement timestamp	*Long*	If there is one, Unix timestamp, in s, of the like

How can we create a tweet similarity?

Bert

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

Text is codified into tokens

```
"[CLS] Not really a fan of rap but Saweetie be givin'me THAT vibe [UNK] [SEP]"
'101 16040 30181 169 10862 10108 35562 10473 74666 23203 10400 10347 38356 15478 112 10911 157 58132 11090 13956 11044 100 102'
```

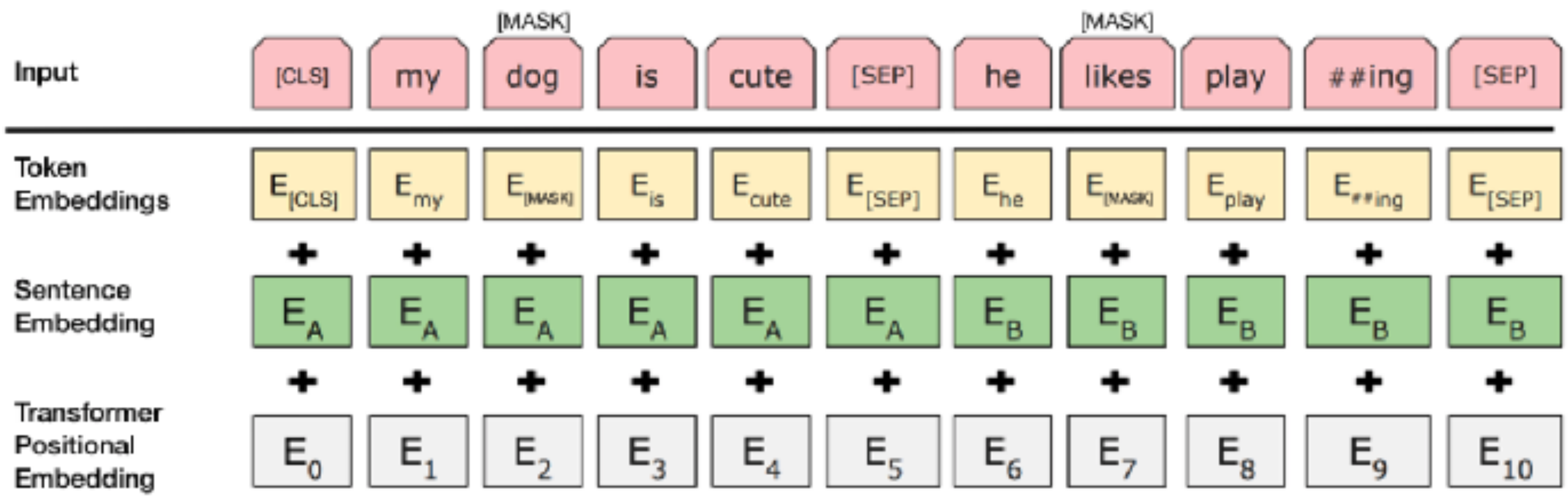
How can we create a tweet similarity?

Bert

BERT (language model)

From Wikipedia, the free encyclopedia

Bidirectional Encoder Representations from Transformers (BERT) is a technique for NLP (Natural Language Processing) pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google.[1][2] Google is leveraging BERT to better understand user searches.[3]



https://github.com/google-research/bert

```
'101 10808 10173 61644 12387 11132 13474 10108 31206 37715 10251 117 10462 10571 32719 119 14120 131 120 120 188 119 11170
120 12428 63051 11537 10138 11369 11373 11259 10477 14120 131 120 120 188 119 11170 120 170 11396 11779 54889 14703 10729
11281 11166 10731 102'
```

```
US confirms second case of coronavirus, 50 under investigation. https : / / t. co / 76krNuL9Ev https : / / t. co / b8VGCY2I5S [SEP]
```

Some similar tweets?

```
(1.0000002, '[CLS] US confirms second case of coronavirus, 50 under investigation. https : / / t. co / 76krNuL9Ev https : / / t. co / b8V
(0.8663348, '[CLS] China coronavirus : Death toll rises as disease spreads https : / / t. co / Cko921fTRZ [SEP]')
(0.8271704, '[CLS] Las autoridades chinas confirman el primer caso de curación del nuevo coronavirus https : / / t. co / yY1dx6ATYR [SEP]
(0.8174852, "[CLS] Inside President Trump's high - stakes impeachment defense effort https : / / t. co / qWHQ8mAfQW https : / / t. co / 
(0.7904908, '[CLS] NEW PEER NEWS PIECE!.. Person of the Year : The French Worker on Strike. https : / / t. co / EGaxAvhh7D https : / / t.
(0.7811252, '[CLS] AB6IX make dumplings & amp ; tteokguk by hand for Lunar New Year!. https : / / t. co / hYGrYmHBKl https : / / t. co /
(0.76791394, "[CLS] Alaska health officials monitoring Coronavirus, deferring to CDC's guidance on when additional measures may be needed
-----
```

UNIVERSITAT DE BARCELONA

How to improve content econding?

Keywords are **not appropiate** for representing content, due to **polysemy**, **synonymy**, **multi-word concepts**,….

Keyword-based Models

doc1
AI is a branch of computer science

doc2
the 2011 International Joint Conference on **Artificial Intelligence** will be held in Spain

doc3
apple launches a new product…

Items

User Profile

artificial	0.02
intelligence	0.01
apple	0.13
AI	0.15
…	…

Multi-words Concepts

Synonyms

apple

Polysemy

NLP methods are needed for the elicitation of user interests

Richer representations

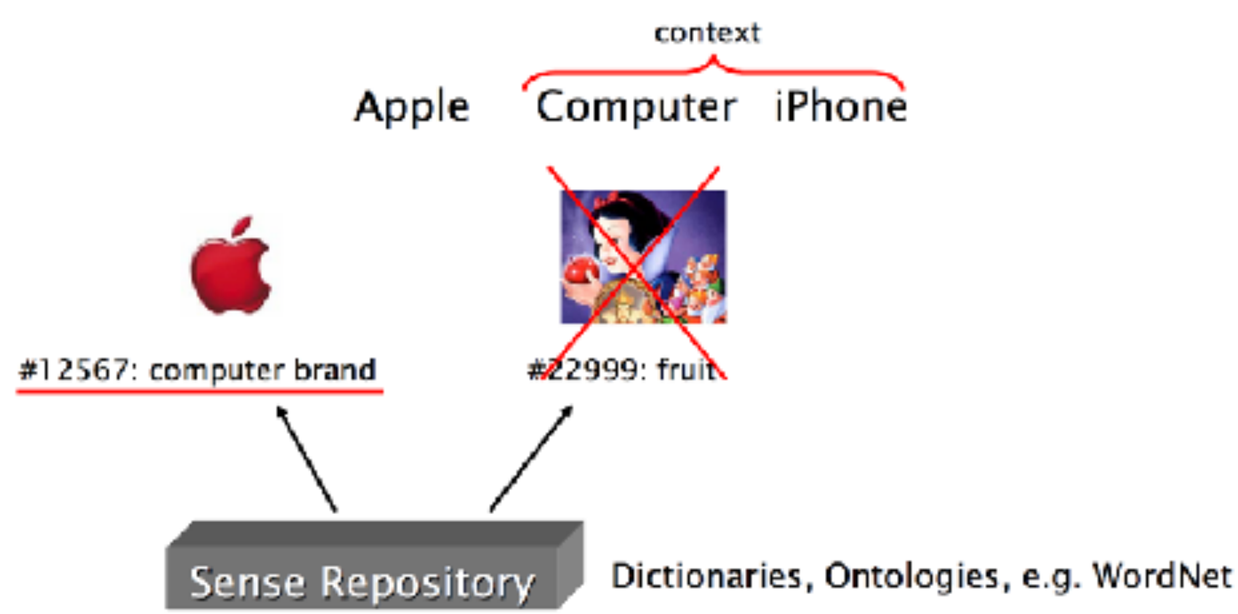
- Semantic Analysis

  - **Semantics:** concept identification in text-based representations through advanced NLP techniques ->“beyond keywords”

  - **Personalization:** representation of user information needs in an effective way -> “deep user profiles”

Sematic Analysis using Ontologies

- Word sense Disambiguation (WSD) -> From words to meanings
    - WSD selects the proper meaning (sense) for a word in a text by taking into account the context in which that word occurs

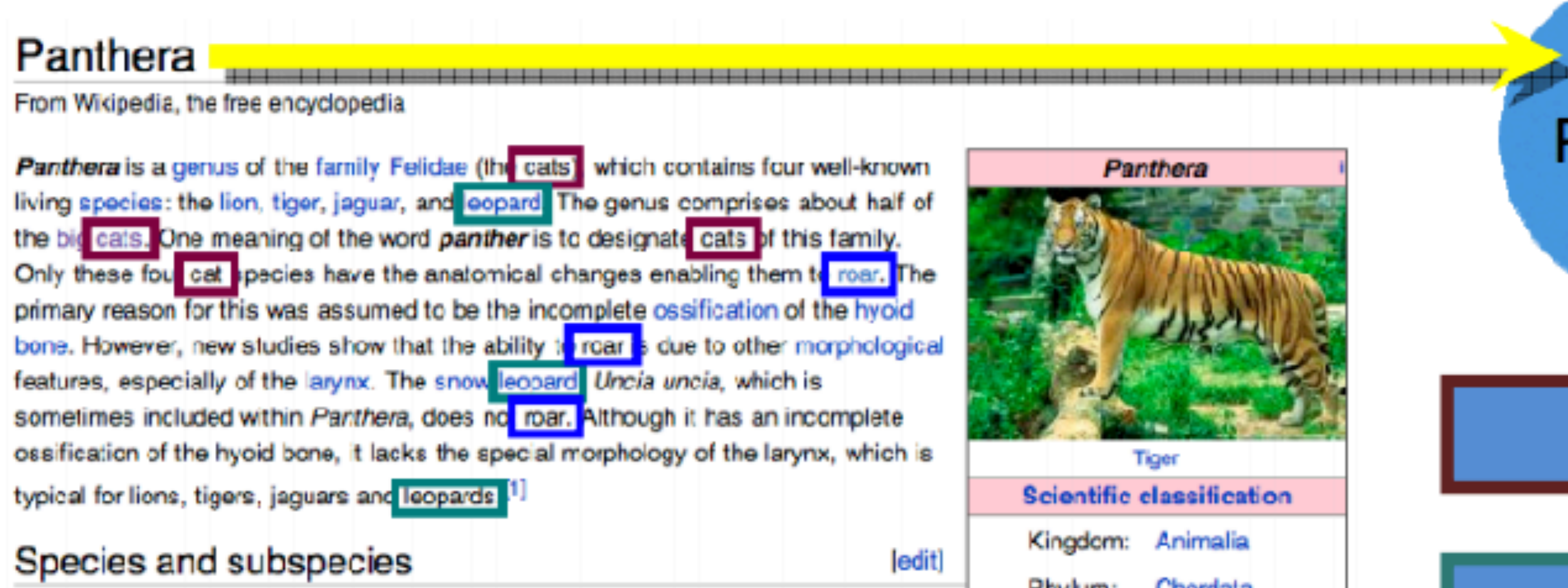


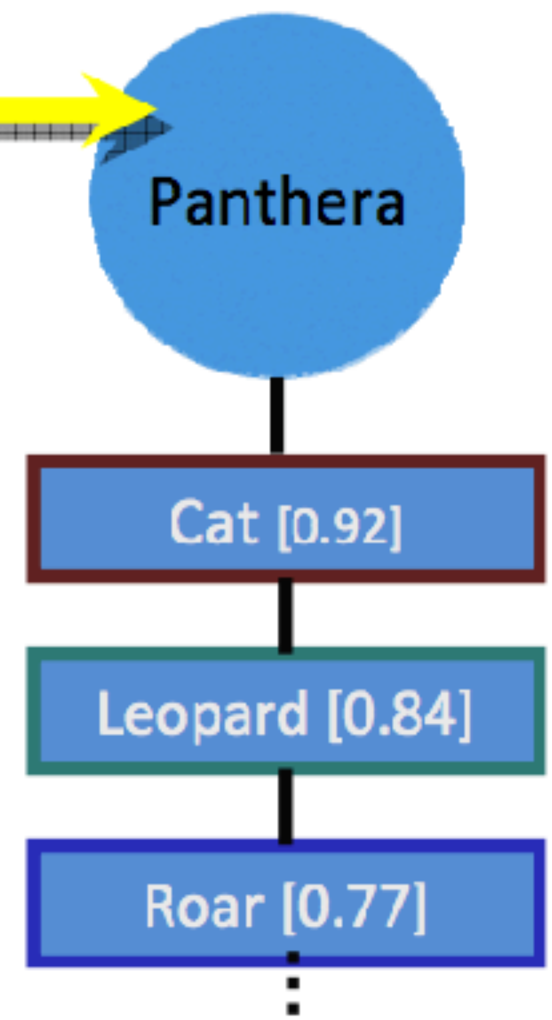
A content-collaborative recommender that exploits WordNet-based user profiles for neighborhood formation | 181 | 2007
M Degemmis, P Lops, G Semeraro
User Modeling and User-Adapted Interaction 17 (3), 217-255

Sematic Analysis using Enclyclopedic Knowledge Sources

Wikipedia is viewed as an **ontology** - a collection of ~**1M** concepts

Every Wikipedia article represents a **concept**

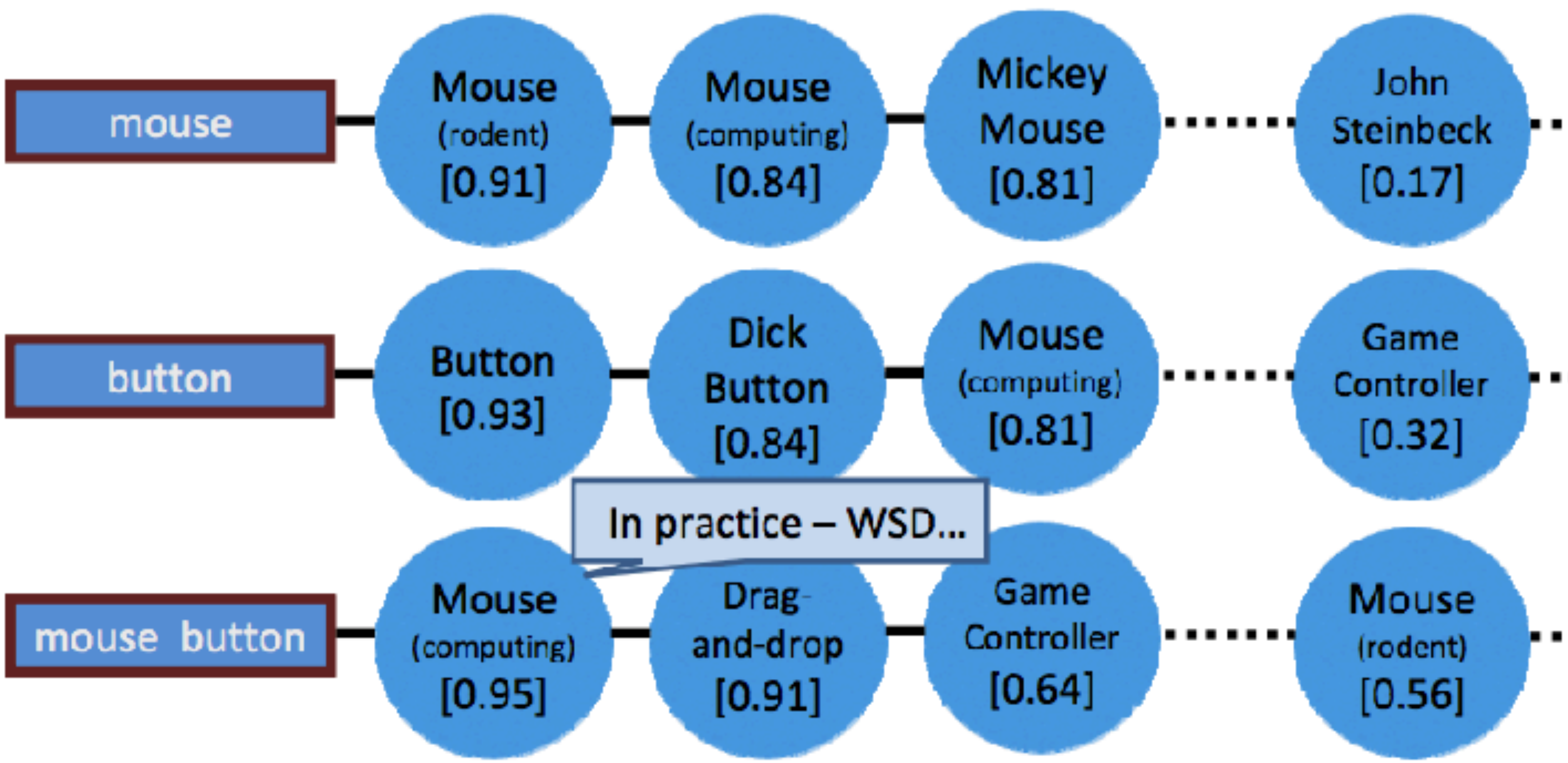




Article **words** are **associated** with the **concept** (TF-IDF)

Sematic Analysis using Enclyclopedic Knowledge Sources

The semantics of a text fragment is the average vector (centroid) of the semantics of its words



Matrix Facorization

- Latent Semantic Analysis
- Latent Dirichlet Allocation

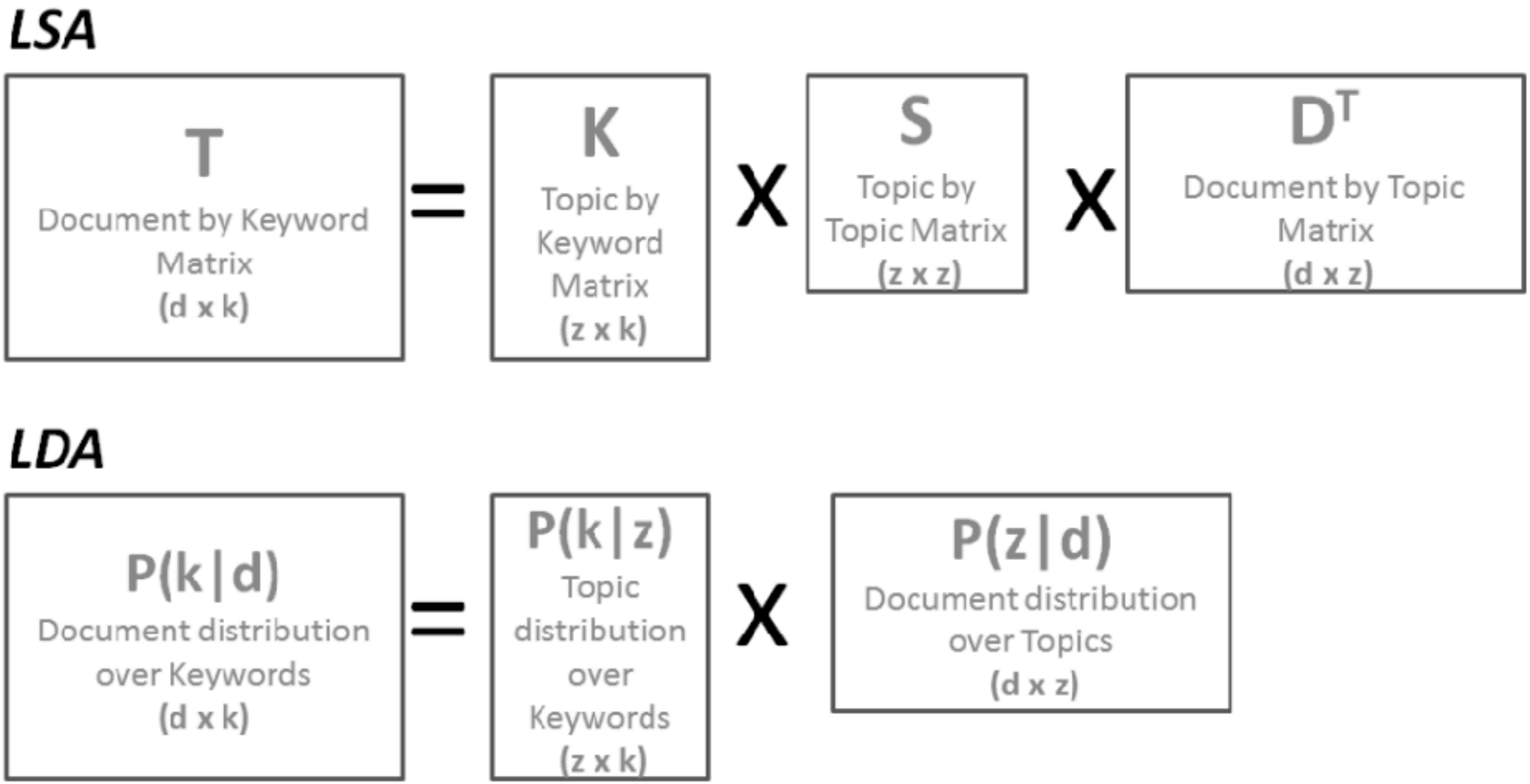
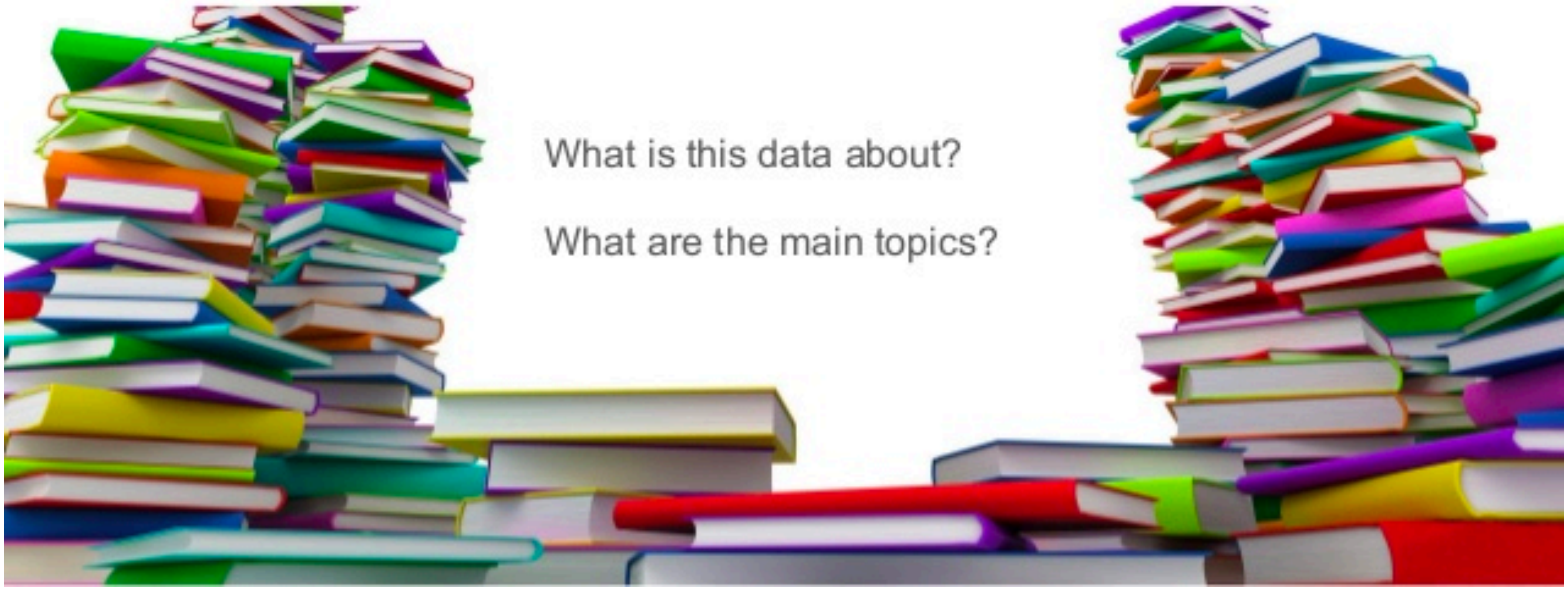


Fig. 2: Matrix decomposition for LSA and LDA.

Topic Modeling

A simple way to analyze topics of large text collections (corpus).



Latent dirichlet allocation
DM Blei, AY Ng, MI Jordan
Journal of machine Learning research 3 (Jan), 993-1022

17706 2003

UNIVERSITAT DE BARCELONA

Topic 1		Topic 2		Topic 3	
term	**weight**	**term**	**weight**	**term**	**weight**
game	0.014	space	0.021	drive	0.021
team	0.011	nasa	0.006	card	0.015
hockey	0.009	earth	0.006	system	0.013
play	0.008	henry	0.005	scsi	0.012
games	0.007	launch	0.004	hard	0.011

Latent dirichlet allocation
DM Blei, AY Ng, MI Jordan
Journal of machine Learning research 3 (Jan), 993-1022

UNIVERSITAT DE BARCELONA

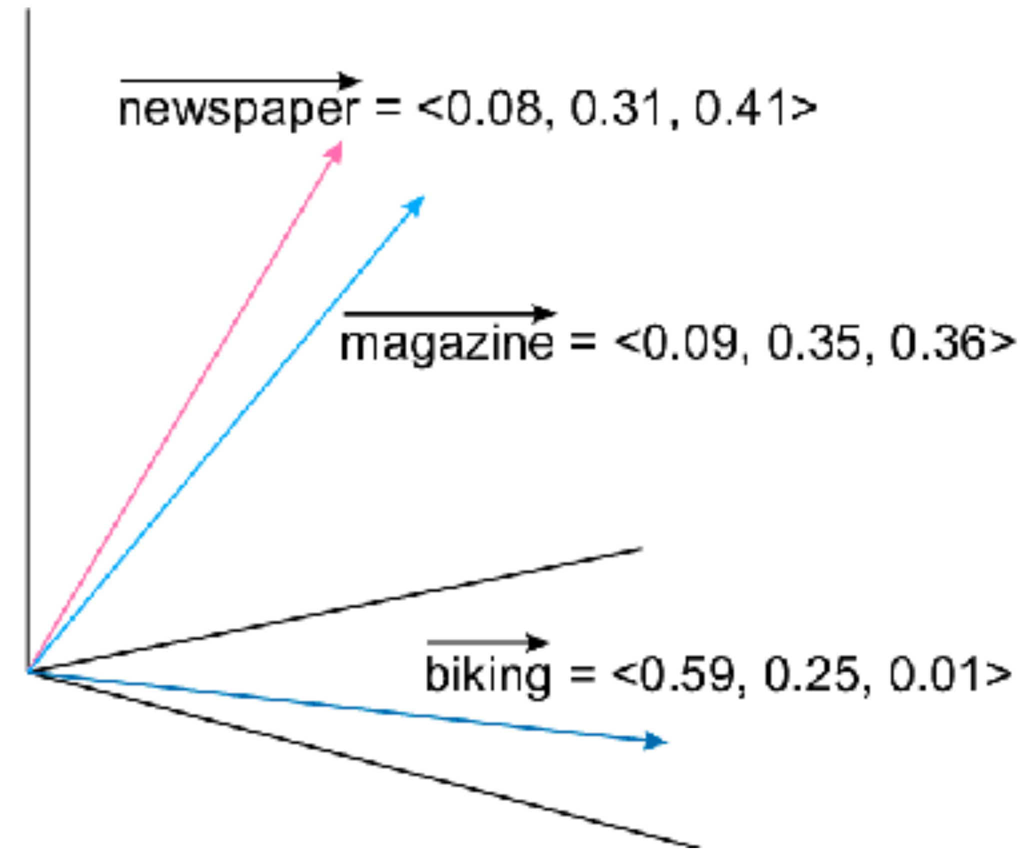
word2vec

(WATER - WET) + FIRE = FLAMES

(PARIS - FRANCE) + ITALY = ROME

(WINTER - COLD) + SUMMER = WARM

(MINOTAUR - MAZE) + DRAGON = SIMCITY



: Target Word

: Context Word

c=0 The cute **cat** jumps over the lazy dog.

c=1 The *cute* **cat** *jumps* over the lazy dog.

c=2 *The* *cute* **cat** *jumps* *over* the lazy dog.

Text Similarities : Estimate the degree of similarity between two texts

Adrien Sieg Follow
Jul 5, 2018 · 29 min read ★

Note to the reader: Python code is shared at the end

We always need to **compute the similarity in meaning between texts.**

- **Search engines** need to model the relevance of a document to a query, beyond the overlap in words between the two. For instance, **question-and-answer** sites such as Quora or Stackoverflow need to determine whether a question has already been asked before.
- In **legal matters**, text similarity task allow to mitigate risks on a new

https://medium.com/@adriensieg/text-similarities-da019229c894