INSIGHT

Spark MLlib
Recommender Systems

Disciplina de Garimpagem de Dados [12/12/2017]

INSIGHT
DEPARTAMENTO DE COMPUTAÇÃO
UNIVERSIDADE FEDERAL DO CEARÁ

A brief introduction to Recommender Systems

Recommender Systems

- Recommender Systems - RSs - are software tools and techniques providing suggestions for **items** to be used by a **user**
- **Item** is general term used to denote what the system recommends
- The system can be **personalized** and **non-personalized**
- RSs are powerful tools to cope with the **information overload problem**
- The goal is to **predict** a rating a **user *u*** will give to a **new item *i***
    - Can be seen as a prediction problem

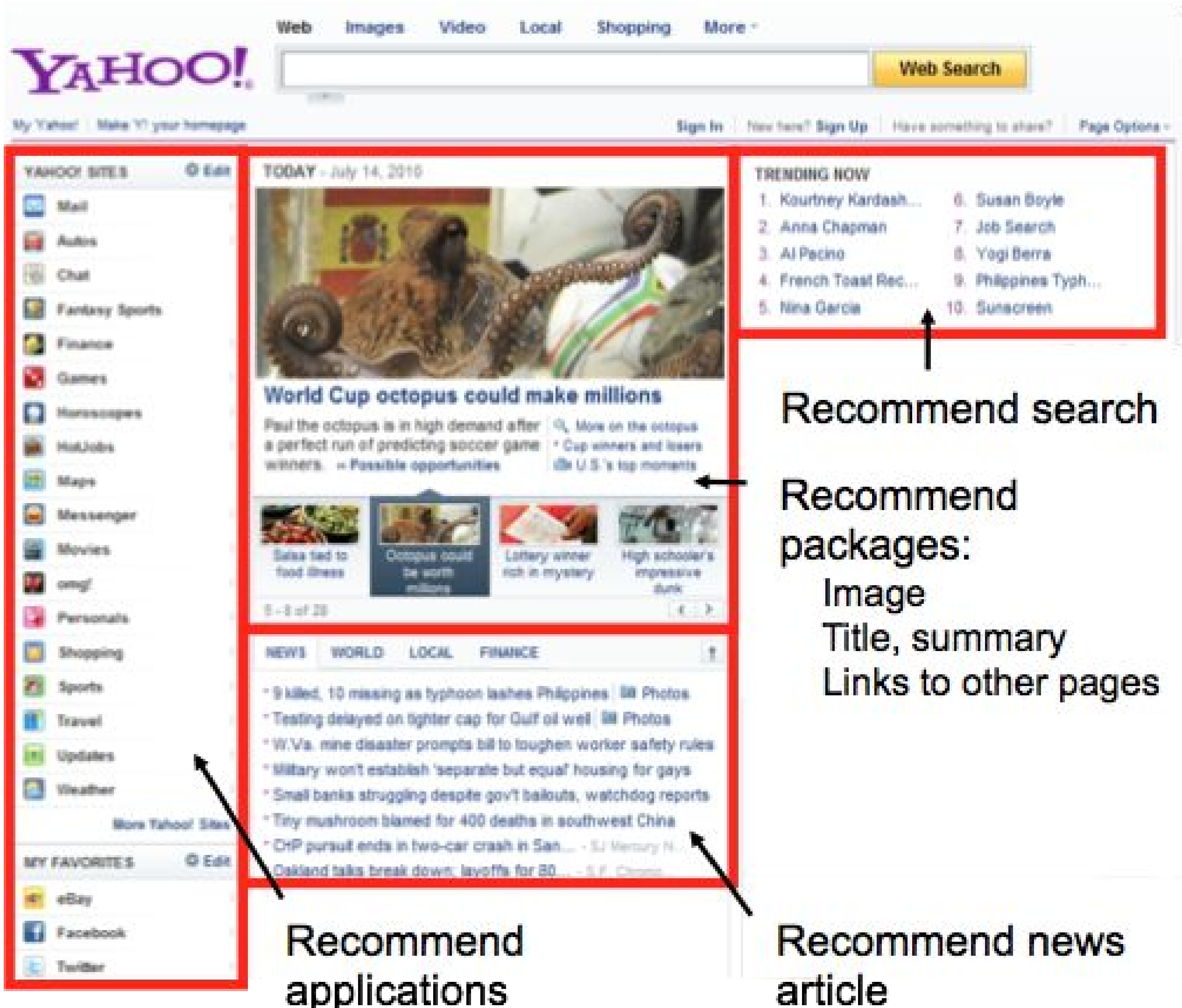
$$\hat{r}_{ui}$$

Recommender System Handbook. F.Ricci, et al. 2011

5

Recommender Systems Applications

- News
- Places
- Movies
- Products
- Musics



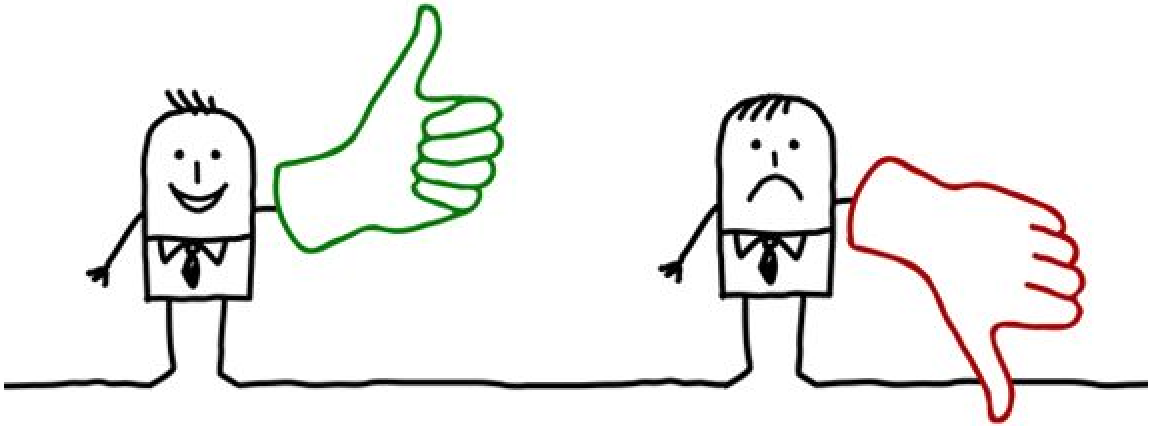
Utility Matrix

Two classes of entities **users** and **items**

Filme	Alice	Bob	Carol	Dave
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

Utility Matrix

- System can ask users to rate items: **explicit feedback**
- Inference from users` behavior: **implicit feedback**
  - clicks, views, etc



Recommender Systems Techniquest

- **Content-based**. The system learns to recommend items that are similar to the ones that the user liked in the past
- **Collaborative Filtering.** Recommendation is generated to an active user based on items that other users with similar tastes liked in the past
- **Demographic.** This type of system recommends items based on the demographic profile of the user: language, country, age, etc.
- **Knowledge-based.** Recommendation based on inference about the users preferences about how certain item features meet users needs

Recommender Systems Techniques

- **Community-based.** Recommendations based on the social connections of the users: "Tell me who your friends are, and I will tell you who you are"
- **Hybrid Recommender.** Combines RSs trying to use advantage of a RS to overcome a limitation of another one
- We focus here on Content-based and Collaborative Filtering

Content-based Recommender System

Content-based recommendations

- This technique is based on **item profile**
  - Records representing important **characteristics**
  - **Movies**: actors, director, year of the movie, genre, etc.
- When features are not immediately apparent
  - E.g. **Documents**, images
  - TF-IDF, bag of words
- Item profile can be represented as a **feature vector**

Movie	Drama	Action
Love at last	1	0
Swords vs. Karate	0	1

Content-based recommendations

Filme	Drama	Action
Love at last	1	0
Romance forever	1	0
Cute puppies of love	1	0
Nonstop car chases	0	1
Swords vs. karate	0	1

Drama

Action

Content-based recommendations

- **User profile** is created using the same features of **item profile**

Movie	Drama	Action
Love at	1	0
Swords	0	1

	Drama	Action
Alice	5	0
Dave	0	4

- Recommendations are generated using similarity functions between the feature vectors
  - Cosine similarity
  - Jaccard similarity
  - etc

Collaborative Filtering RS

Collaborative Filtering RS

- Focus on similarity of the user ratings of two items
- Instead of **item profile**, we use the **row** of the utility matrix

Filme	Alice	Bob	Carol	Dave
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

Collaborative Filtering RS

- Collaborative filtering RSs are grouped into:
  - **Neighborhood** method
    - user-item ratings are directly used to predict ratings for new items
  - **Model-based** method
    - uses ratings to learn a predictive model
    - Model user-item interactions with factors representing latent characteristics

Neighborhood methods

- User-based neighborhood
  - Predict the rating *r^_ui* of a user ***u*** to a new item ***i*** using ratings given to ***i*** by users most similar to ***u***
  - *k*-NN is used to find the most similar users *N_i(u)*
  - *w_uv* similarity between user **u** and user **v**

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|}.$$

Neighborhood methods

- Item-based neighborhood
  - Predict the rating *r^_ui* of a user ***u*** to a new item ***i*** using ratings given to similar items to ***i*** given by ***u***
  - *k*-NN is used to find the most similar items rated by ***u*** - *N_i(u)*
  - *w_ij* similarity between item **i** and item **j** based on ratings

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{j \in N_u(i)} |w_{ij}|}.$$

Model methods

- Uses ratings to learn a predictive model
- Model user-item interactions with factors representing latent characteristics
- Singular Value Decomposition (SVD), **Latent models**, etc
- Most successful latent factor models are based on *matrix factorization*
  - Characterizes both items and users by vectors of factors inferred from item rating patterns

Matrix Factorization Model

- 2 dimension
- female- versus male-oriented
- serious versus escapist
- Gus tends to **love** Dumb and Bumber and **hate** The Color Purple

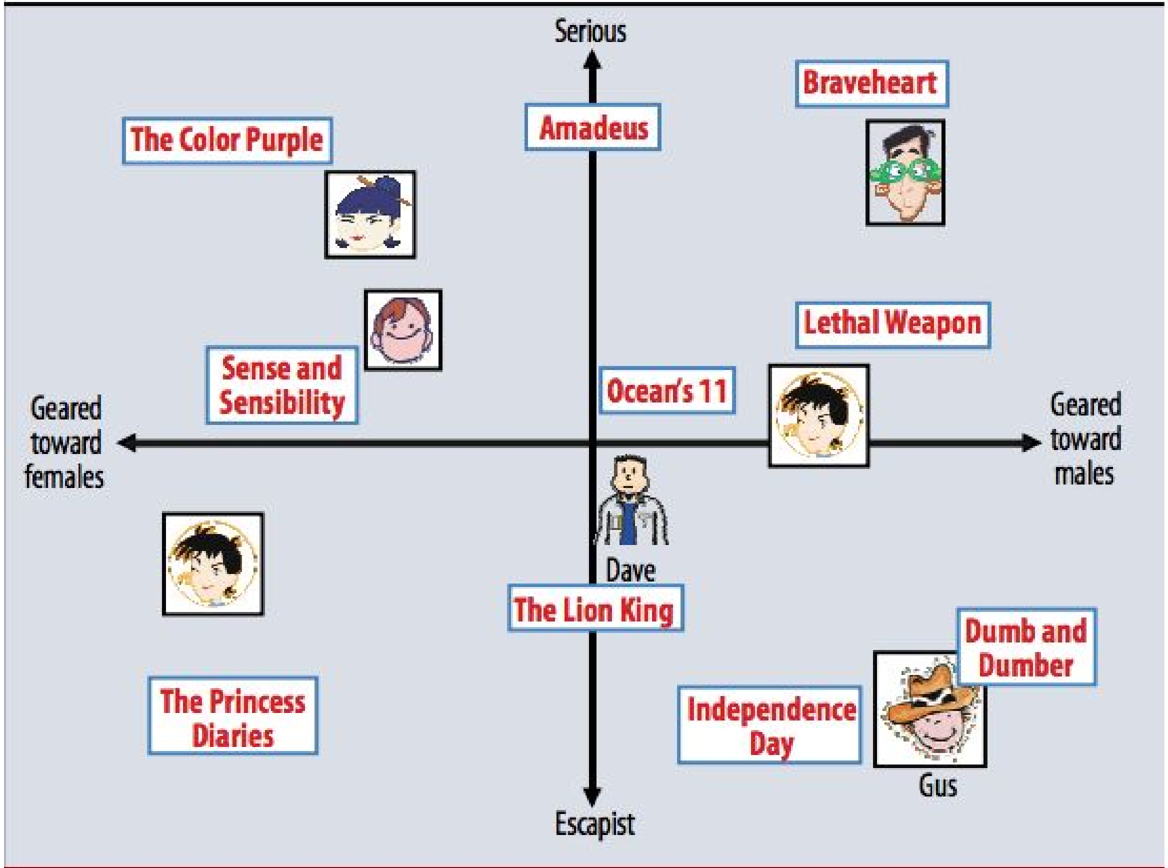


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Basic Matrix Factorization Model

- MF model map users and items to a joint latent factor space of *f* dimensions
- item *i* is associated with a vector *q_i* of *f* dimension
  - elements in *q_i* measure the extent to which the item possesses those factors, positive or negative
- user *u* is associated with a vector *p_u* of *f* dimension
  - *p_u* measure the extent of interest the user has in items that are high on the corresponding factor, positive or negative

$$\hat{r}_{ui} = q_i^T p_u.$$ user-item interaction

Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in *Computer*, vol. 42, no. 8, pp. 30-37, Aug. 2009

Basic Matrix Factorization Model

- The learning model
- Minimize the regularized squared error on the set of known ratings
- *r_ui* known rating (training set)

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

Basic Matrix Factorization Model

- Techniques to minimize the previous equation
  - Stochastic gradient descent
    - Update parameters by a magnitude proportional to gamma
    - Easy implementation and fast running time
  - **Alternating least squares (Spark MLlib)**
    - Convert the equation into a quadratic optimization problem
    - Rotate between fixing q_i`s and p_i`s
    - Allow parallelization
      - q_i is independent of the other item factors
      - p_u is independent of the other user factors
    - Favor for system centered on implicit data

Alternating least squares on Spark MLlib

Alternating least squares - ALS

- **Basics**
  - **Input**: itemCol, userCol, ratingCol,

    rank number of latent factors

    regParam reguralization param

    implicitPrefs explicit / implicit

    alpha confidence for implicit feedback

    nonnegative use or not non-negative constraint
  - **Data Format:** LibSVM
  - **Output:** predictionCol