Big Data Computing 2020
Recommender Systems

SAPIENZA
UNIVERSITÀ DI ROMA

Donato Crisostomi

Master Degree in
Computer Science
Sapienza, University of Rome

A.Y. 2019 - 2020

Motivation

- recent rise of web services offering all kind of goods;

Motivation

- recent rise of web services offering all kind of goods;
- too much information may overwhelm the user;

Motivation

- recent rise of web services offering all kind of goods;
- too much information may overwhelm the user;
- need to narrow the proposed content;

Motivation

- recent rise of web services offering all kind of goods;
- too much information may overwhelm the user;
- need to narrow the proposed content;
  - greater income;

Motivation

- recent rise of web services offering all kind of goods;
- too much information may overwhelm the user;
- need to narrow the proposed content;
  - greater income;
  - stand out from competitors;

Formalization

recommender systems are algorithms aimed at suggesting relevant items to users.

Formalization

recommender systems are algorithms aimed at suggesting relevant items to users.

- **content-based**: create items and users profiles, embed them in a numerical feature space, and then suggest to the user the items which are nearest to him;

Formalization

recommender systems are algorithms aimed at suggesting relevant items to users.

- **content-based**: create items and users profiles, embed them in a numerical feature space, and then suggest to the user the items which are nearest to him;
- **collaborative-filtering**: ignore content and rely only on user-item ratings
  - **user-based**: user-user similarity;
  - **item-based**: item-item similarity;

Formalization

recommender systems are algorithms aimed at suggesting relevant items to users.

- **content-based**: create items and users profiles, embed them in a numerical feature space, and then suggest to the user the items which are nearest to him;
- **collaborative-filtering**: ignore content and rely only on user-item ratings
  - **user-based**: user-user similarity;
  - **item-based**: item-item similarity;
- **hybrid**: leverage both approaches.

Sequential recommender systems

Recommender systems have recently taken in consideration sequential dynamics.

- capture patterns in the sequence of actions users perform;
- ignore time, model the **order** of the actions;
- challenging task as possible sequences grow exponentially;
- **Markov-Chain models** condition the next action only on few previous ones;
- Neural architectures can be used;

In general feedback is not always given explicitly, so other approaches try to exploit **implicit feedbacks** like clicks or purchases.

Recommending games

In this project we will see how recommender systems can be leveraged to suggest **games**.

Dataset

The dataset contains information regarding both games and user-game reviews, separated in two tables **steam_reviews** and **steam_games**:

steam_reviews:

- username
- user_id
- product_id
- text
- date
- found_funny
- hours_played

steam_games:

- title
- id
- developer
- genres
- metascore
- price
- publisher
- release_date
- specs
- tags

Dataset

We will use two models, plus a naive baseline:

- **MF model**, which is a collaborative-filtering approach;
- **RNN-based**, which is a sequential recommendation approach;

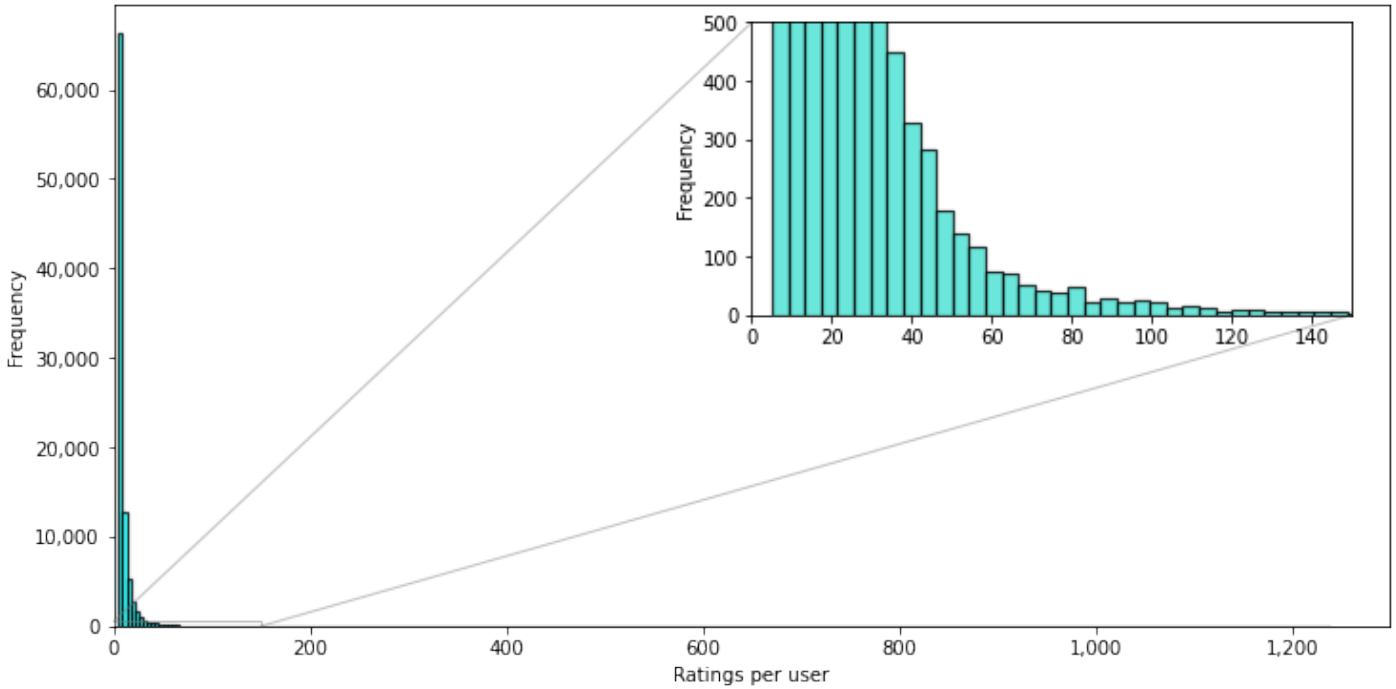
As both don't require item profiles, we will not use the game features.

Final set of features:

- **user_id**: reviewer id;
- **product_id**: reviewed game id;
- **text**: content of the review;
- **date**: date of the review;

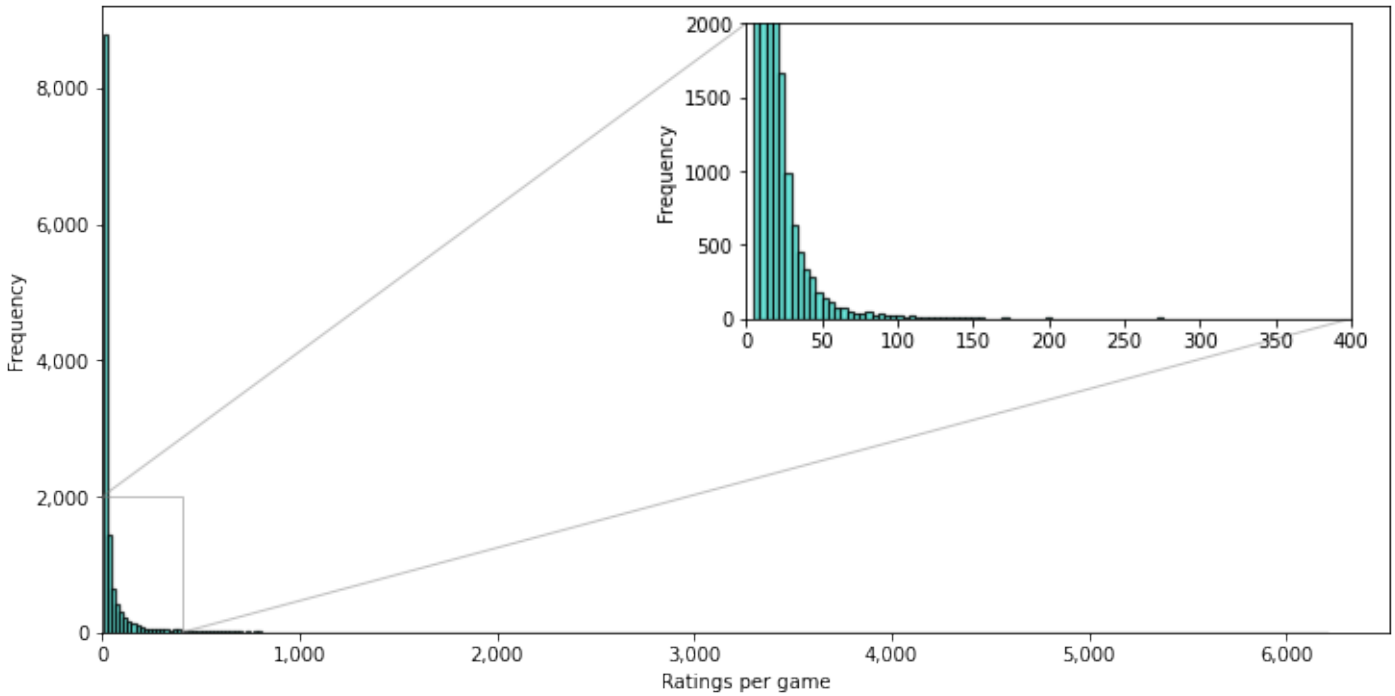
Data Analysis: ratings per user

The distribution of ratings per-user follows a power-law: most of the users reviewed few games, while only few users have reviewed a significant number of games.



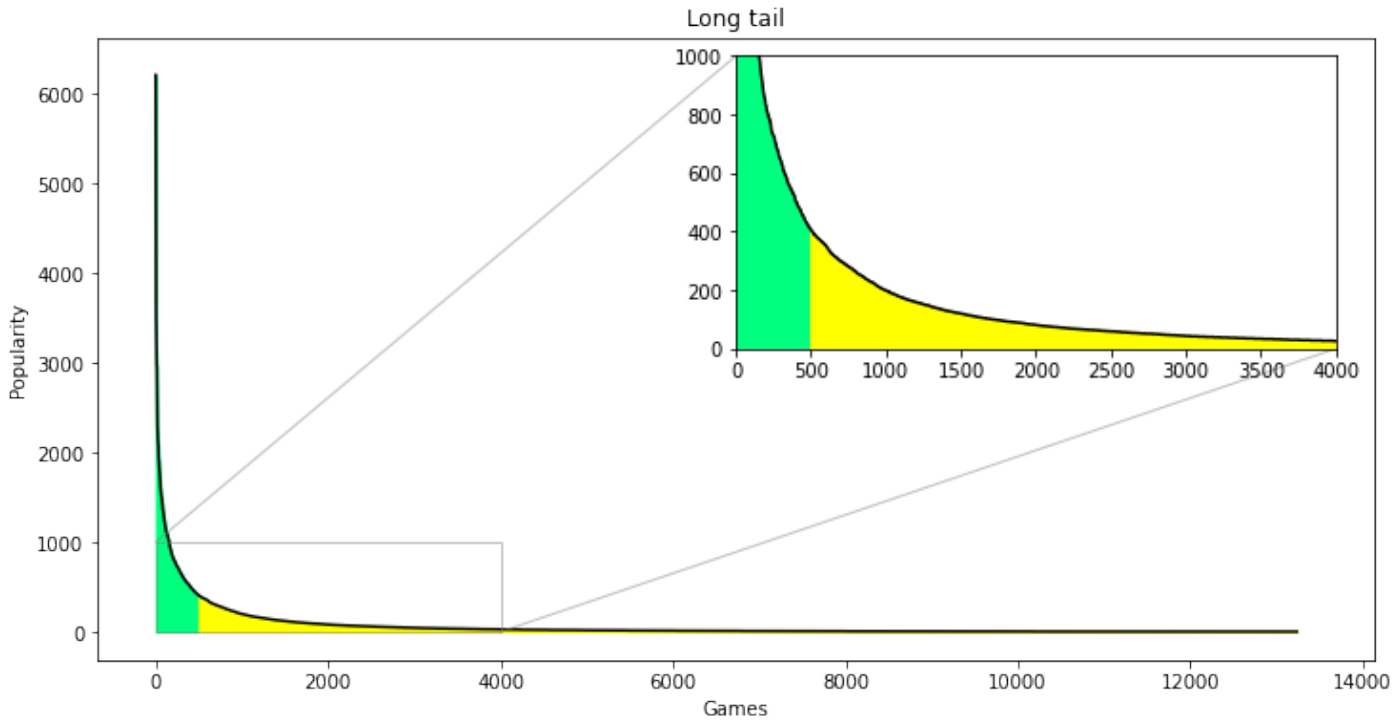
Data Analysis: ratings per game

The same thing happens for games, few games received a large amount of reviews, while most games have few.



Data Analysis: long tail

The distribution shows a long tail which amounts for a significant part of the catalogue, so a good recommender system should be able to recommend less famous games, even if it is in fact harder.



Adding sentiment

- explicit ratings matrix required;
- implicit feedback approaches may waste useful information from reviews;
- turn text reviews into numerical ratings;

Adding sentiment

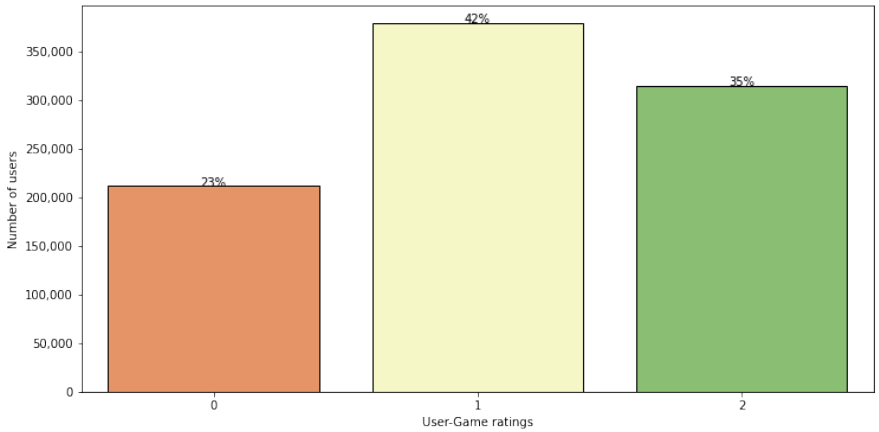
- explicit ratings matrix required;
- implicit feedback approaches may waste useful information from reviews;
- turn text reviews into numerical ratings;

For this task, we will use a pretrained **sentiment analyzer** from StanfordNLP, so every text review will be mapped to a value $r \in \{0, 1, 2\}$, meaning negative, neutral and positive respectively.

Adding sentiment

- explicit ratings matrix required;
- implicit feedback approaches may waste useful information from reviews;
- turn text reviews into numerical ratings;

For this task, we will use a pretrained **sentiment analyzer** from StanfordNLP, so every text review will be mapped to a value $r \in \{0, 1, 2\}$, meaning negative, neutral and positive respectively.



Matrix Factorization

We will use a constant model as a baseline

- constant in the sense that it does not depend on the user in input;
- without information regarding the user, the safest bet is to suggest most popular games.

The first real model employs a **Latent Factor** collaborative filtering approach.

Latent factor models

Latent factor models are statistical models that relate a set of observable variables (so-called manifest variables) to a set of latent variables.

In our case we want to predict user ratings by representing both items and users with a number of hidden factors inferred from observed ratings.

Matrix Factorization

- we assume there exists an unknown low-dimensional representation of users and items;
- **Matrix Factorization** obtains these lower-dimensional representations directly from data.

In general, we want to infer the rating $r_{u,i}$ of user u to item i.

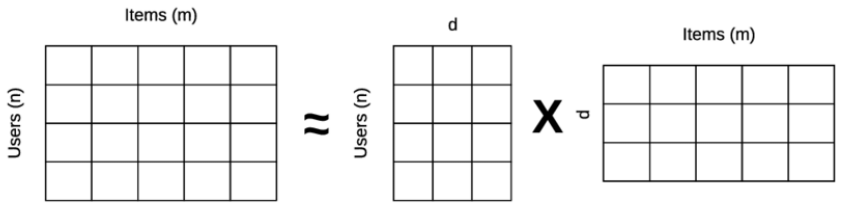
- map both items and users to a **joint latent factor** d-dimensional space, so
  - each user will be represented by $\mathbf{x}_u \in \mathbb{R}^d$,
  - each item will be represented by $\mathbf{w}_i \in \mathbb{R}^d$;
- estimate $r_{u,i}$ by applying the dot product

$$\hat{r}_{u,i} = \mathbf{x}_u^T \cdot \mathbf{w}_i = \sum_{j=1}^{d} x_{u,j} w_{j,i} \tag{1}$$

- recommend to user u the k items for which the estimate is maximum.

Matrix Factorization

Need a way to map users and items to these latent factor vectors.



- Approximate R with the product of two matrices $X \in \mathbb{R}^{m \times d}$ and $W \in \mathbb{R}^{d \times n}$
- Equivalent to minimizing

$$\mathcal{L}(X, W) = \sum_{u,i \in D} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i \right)^2 \tag{2}$$

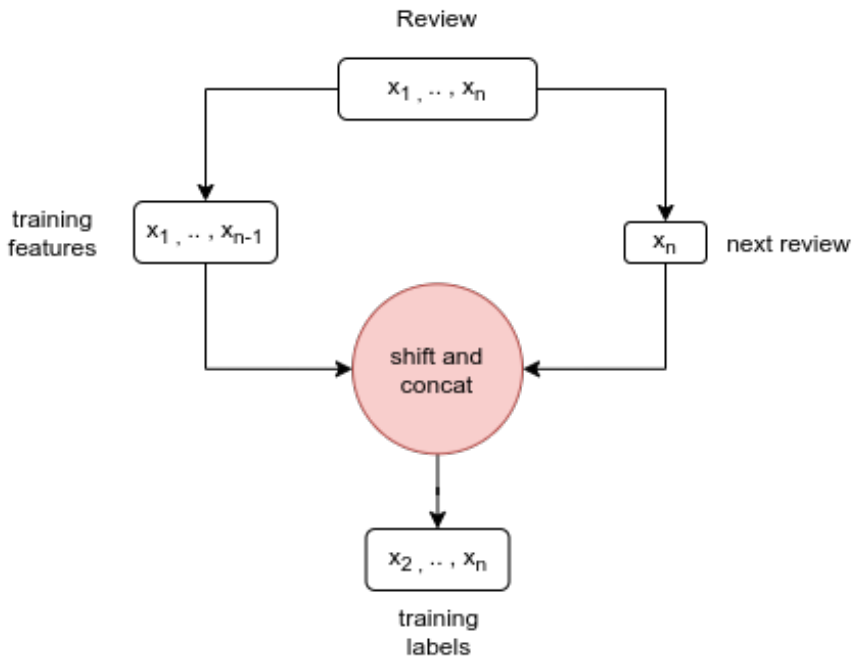
- Optimization problem can be solved in 2 ways
  - **SGD**: iterative method which tries to minimize the loss function by descending its gradient;
  - **ALS**: alternately fix one latent vector and update the other one, making the loss function convex;

Sequential Recommender System

Transform the dataset to obtain for each user a vector

$$\mathbf{x} = (x_1, \ldots, x_n)$$

- Train the model to predict next game review given the previous ones, or equivalently x_i given $x_1, \ldots, x_{i-1}$;
- use the first $n-1$ reviews $(x_1, \ldots, x_{n-1})$ as input and train the model to predict the input shifted by 1 position $(x_2, \ldots, x_n)$.



LSTM

Recurrent Neural Networks summarize the context of a certain token m with a recurrently updated vector

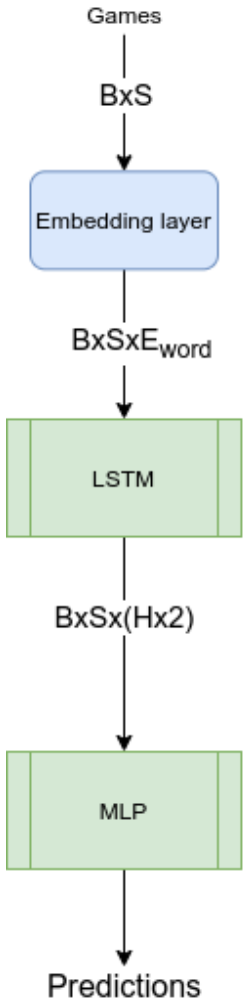
$$\vec{h}_m = g(\vec{x}_m, \vec{h}_{m-1}), \quad m = 1, 2, \ldots, m$$

where $\vec{x}_m$ is the vector embedding of the token w_m and g defines the recurrence.

- often fail to capture long-time dependencies;
- LSTMs often used instead;
- more complex recurrence, in which a memory cell goes through a series of gates, in fact avoiding repeated applications of non-linearity.
- bidirectionality would allow the model to cheat;
- sequences must be padded;

1. each input sequence **x** is embedded by a **word embedding layer**;
2. a *LSTM* encoder takes as input the embedded sequence and returns a **dynamic representation** of each game and its context;
3. the hidden representation is given to a **Multi Layer Perceptron** to map each game representation to the games space.
4. Cross entropy is used as loss function

$$\mathcal{L} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$



Legend	
E_{game}	game embedding dim
S	Sequence length
H	LSTM hidden dimension
B	batch size

Matrix Factorization evaluation

Root Mean Squared Error quantifies how much much the predictions are far from the gold truth.

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{N}} \tag{3}$$

Precision at k instead is a measure of how many of the first k recommended documents are in the set of true relevant documents averaged across all users.

$$P(k) = \frac{1}{M}\sum_{i=0}^{M-1}\frac{1}{k}\sum_{j=0}^{\min(Q_i,k)-1} rel_{D_i}(R_i(j)) \tag{4}$$

As $P@k$ doesn't account for the order of the recommendations, to consider it we must use $NDCG@k$

$$NDCG(k) = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{IDCG(D_i, k)} \sum_{j=0}^{n-1} \frac{rel_{D_i}(R_i(j))}{\log(j+2)} \tag{5}$$

where

$$n = \min\left(\max\left(Q_i, N_i\right), k\right) \tag{6}$$

$$IDCG(D, k) = \sum_{j=0}^{\min(|D|,k)-1} \frac{1}{\log(j+2)} \tag{7}$$

Matrix Factorization results

	P@5	NDCG@5	RMSE
Popularity	$4e^{-3}$	$9e^{-3}$	//
Matrix Factorization	$3e^{-5}$	$9e^{-5}$	1.125

Problem challenges:

- sentiment analysis still an open research field;
- severe error pile-up with the error produced by the recommender system;
- game reviews may be particularly difficult for the analyzer to get right;

Metrics for the sequential approach

Hit@10 has been used for the sequential model, which is a **top**-n metric that counts the fraction of times that the ground-truth next item is among the top 10 recommended items.

$$\frac{\#\text{ hits}}{\#\text{ users}} \tag{8}$$

- model obtains a $H@5$ of 0.503;
- obtained accuracy would be low in a typical classification setting with few classes;
- model must be able to discriminate among $\approx 13k$ classes

Sequential results

Random baseline would obtain

$$\mathbf{Pr}\{hit\} = \sum_{i=1}^{10} \mathbf{Pr}\{pred_i \text{ is correct}\} \tag{9}$$

$$= \sum_{i=1}^{10} \frac{1}{\#\text{ classes}} = 10 \cdot \frac{1}{13000} \approx 7e^{-5} \tag{10}$$

Conclusions

We have seen two very different approaches to recommender systems, neither of them reached state of the art.

- The MF approach is too naive and doesn't exploit all the features;
- the LSTM approach has few intrinsic flaws, for example the LSTM predicting a sequence of games where the same game is often repeated;
- not easy to carve such constraints in the model.

There are some improvements that could be tried:

- add content based features to both approaches, for example obtaining contextualized embeddings from the text reviews;
- try adding a sequence scorer on top of the LSTM (e.g. a CRF) to help assess the quality of a sequence of tags as a whole.

Thank you for your attention