

From Implicit Preferences to Ratings

Video Games Recommendation based on Collaborative Filtering

Rosária Bunga¹, Fernando Batista^{1,2}, Ricardo Ribeiro^{1,2}¹ Iscte - Instituto Universitário de Lisboa, Lisboa, Portugal² INESC-ID Lisboa, Portugal

{rosaria_patricia_bunga, fernando.batista, ricardo.ribeiro}@iscte-iul.pt

Motivation and goals

Propose a videogame-oriented recommendation system

- Provide suggestions based on a **user gaming history** and the **tastes of similar users**
- Study the performance of **collaborative filtering algorithms** over Steam platform, containing a considerable amount of data
- Understand how to use implicit data to infer explicit user ratings
- Check if playing time of the users serves as an adequate implicit representation of the user's preferences

Dataset and setup

- Purchase history of Australian users of the video game *Steam* platform (Pathak et al., 2017).
 - For each user, it provides the list of purchased items with a small collection of metadata related to game playing time.

Attribute	Type	Description
user_id	string	User
steam_id	integer	Steam user identification
user_url	string	User URL
item_id	integer	Game identification
item_name	string	Game title
playtime_2weeks	integer	Number of minutes played in the last two weeks
playtime_forever	integer	Total number of minutes played

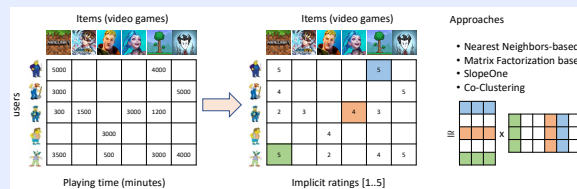
Setup

- We use a filtered dataset
 - Removed: players with less than 30 games, games that were played by less than 10 players, and games that had a playing time of zero minutes
 - **33307** unique users, **6387** unique games, and about **2.8 million** records
 - 80% for training and 20% for testing (Info: 10%, Target: 10%)
- Our experiments use the implementations of the **Surprise library**
 - allows to create and evaluate recommender systems for explicit data
- The performance of the different approaches was evaluated using common metrics used in the evaluation of recommendation systems
 - RMSE, MAE, Precision@k, Recall@k and F1@k

Approaches

From implicit preferences to ratings

- We used the **total playing time** to infer the **explicit ratings**, and to understand the preferences of the users. It is used to quantify the relevance of an item to a specific user, on the assumption that **if a user plays a game for a long time, he likes that game**.
 - For converting implicit ratings into explicit ratings, we grouped the data by user and playing time, and then we grouped the playing times into five bins. Then for each group we computed its rank, using a scale ranging from 1 to 5.



Recommendation Approaches

- Nearest Neighbors-based Algorithms
- Algorithms based on Matrix Factorization
- SlopeOne
- Co-Clustering

Conclusions and future work

- We have tested 7 recommendation systems based on collaborative filtering for the video game domain
- Before training the algorithms, the **users' total playing time** was converted into an explicit rating

Conclusions

- SlopeOne and SVD++ were the best performing recommendation algorithms. Considering computational cost, SlopeOne is the best
- Considering the dataset under study, there is not a significant difference between the performance of the explored algorithms
- It is possible to use playtime to infer explicit ratings for making recommendations in the video game domain and specifically on the Steam platform
- The presented algorithms can produce good recommendations, are easily applicable, and are computationally low demanding approaches, when compared to more complex algorithms.

Future work

- Explore other methods of transforming playing time into explicit ratings; Explore other types of implicit expression of preferences, if available.

Results

- The algorithms show similar performances.

	RMSE	MAE	precision			recall			F1		
			@5	@10	@20	@5	@10	@20	@5	@10	@20
KNNBasic	1.2270	1.0159	0.7374	0.7144	0.7064	0.2866	0.3718	0.3954	0.4128	0.4891	0.5070
KNNWithMeans	1.2483	1.0356	0.7293	0.7043	0.6955	0.2767	0.3595	0.3841	0.4012	0.4760	0.4951
SVD	1.2386	1.0049	0.7619	0.7162	0.7003	0.3367	0.4609	0.5028	0.4670	0.5608	0.5854
SVD++	1.2179	0.9727	0.7757	0.7309	0.7145	0.3397	0.4640	0.5095	0.4725	0.5677	0.5948
NMF	1.2229	1.0011	0.7525	0.7073	0.6920	0.3275	0.4470	0.4882	0.4564	0.5478	0.5725
SlopeOne	1.1977	0.9831	0.7840	0.7336	0.7183	0.3391	0.4535	0.4920	0.4735	0.5605	0.5840
Co-Clustering	1.2354	1.0212	0.7713	0.7288	0.7150	0.3174	0.4229	0.4558	0.4498	0.5352	0.5567

- Overall, **SlopeOne** and **SVD++** achieved the best results for all metrics.
 - Considering the RMSE and MAE, SlopeOne and SVD++ exhibited the best results
 - Concerning precision, SlopeOne achieved the best result
 - Concerning recall, SVD++ performed better, not much different from SVD
 - KNNWithMeans had the worst results
- The distribution of the predicted ratings on the test set is noticeably different for all algorithms, not reflecting the actual distribution of implicit ratings.
- Concerning the average training time, SlopeOne takes about 24 seconds while SVD++ takes about 4904 seconds (~200x), which corresponds to a significant difference. Concerning the testing time, SlopeOne takes about 77 seconds while SVD++ takes 110 seconds, being about 42% slower.

From Implicit Preferences to Ratings

Video Games Recommendation based on Collaborative Filtering

Rosária Bunga, Fernando Batista, Ricardo Ribeiro

Motivation and Goal



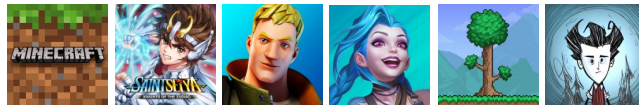
- Propose a **videogame-oriented recommendation system**
 - Provide suggestions based on a **user gaming history** and the **tastes of similar users**
 - Study and compare the performance of **collaborative filtering (CF) algorithms** over a considerable amount of data (Steam video games platform)
- Understand how to use implicit data to infer explicit user ratings
- Check if playing time of the users serves as an adequate implicit representation of the user's preferences



From implicit preferences to ratings

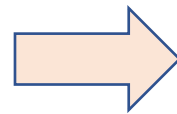
- We used the total playing time to infer the explicit ratings, and to understand the preferences of the users, under the assumption that if a user plays a game for a long time, he likes that game

Items (video games)

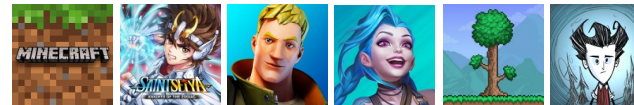


users		5000				4000	
		3000					5000
		300	1500		3000	1200	
				3000			
		3500		500		3000	4000

Playing time (minutes)



Items (video games)



	5				5	
	4					5
	2	3		4	3	
			4			
	5		2		4	5

Implicit ratings [1..5]

Approaches

- Nearest Neighbors-based
- Matrix Factorization based
- SlopeOne
- Co-Clustering

$$\mathbb{R}^{5 \times 6} \times \mathbb{R}^{6 \times 6} = \mathbb{R}^{5 \times 6}$$

| Dataset

- Video games purchase history of users of the *Steam* platform (Pathak et al., 2017)

- For each user, it provides the list of purchased items with a small collection of metadata related to game playing time

Attribute	Description
<i>user_id</i>	User
<i>steam_id</i>	Steam user identification
<i>user_url</i>	User URL
<i>item_id</i>	Game identification
<i>item_name</i>	Game title
<i>playtime_2weeks</i>	Number of minutes played in the last two weeks
<i>playtime_forever</i>	Total number of minutes played

- We have used a filtered dataset:
 - players with less than 30 games, games that were played by less than 10 players, and games that had a playing time of zero minutes were removed
 - 33307 unique users, 6387 unique games, and 2.8 million records

| Setup

- Our experiments use the implementations of the *Surprise library*
 - allows to easily create and evaluate recommender systems that deal with explicit data.
- The performance of the different approaches was evaluated using common metrics used in the evaluation of recommendation systems
 - RMSE, MAE, Precision@k , Recall@k and F1@k
- Training: 80%, testing: 20% (Info: 10%, Target: 10%)

Results

	RMSE	MAE	precision			recall			F1		
			@5	@10	@20	@5	@10	@20	@5	@10	@20
KNNBasic	1.2270	1.0159	0.7374	0.7144	0.7064	0.2866	0.3718	0.3954	0.4128	0.4891	0.5070
KNNWithMeans	1.2483	1.0356	0.7293	0.7043	0.6955	0.2767	0.3595	0.3841	0.4012	0.4760	0.4951
SVD	1.2386	1.0049	0.7619	0.7162	0.7003	0.3367	0.4609	0.5028	0.4670	0.5608	0.5854
SVD++	1.2179	0.9727	0.7757	0.7309	0.7145	0.3397	0.4640	0.5095	0.4725	0.5677	0.5948
NMF	1.2229	1.0011	0.7525	0.7073	0.6920	0.3275	0.4470	0.4882	0.4564	0.5478	0.5725
SlopeOne	1.1977	0.9831	0.7840	0.7336	0.7183	0.3391	0.4535	0.4920	0.4735	0.5605	0.5840
Co-Clustering	1.2354	1.0212	0.7713	0.7288	0.7150	0.3174	0.4229	0.4558	0.4498	0.5352	0.5567

Overall, **SlopeOne** and **SVD++** achieved the best results for all metrics.

| Results

- Overall, **SlopeOne** and **SVD++** achieved the best results for all metrics.
- Concerning the average training time
 - SlopeOne takes about 24 seconds
 - SVD++ takes about 4904 seconds (~200x)
- Concerning the testing time
 - SlopeOne takes about 77 seconds
 - SVD++ takes 110 seconds (42% slower).

| Conclusions

- We have tested 7 collaborative filtering recommendation systems for the video game domain
 - Data from the Steam platform
- Before training the algorithms, the users' total playing time was converted into an explicit rating
 - Playtime can be used to infer explicit ratings, and for recommendations in the video game domain, specifically on the Steam platform
- SlopeOne and SVD++ were the best performing algorithms.
 - Considering computational cost, SlopeOne wins
- There is not a significant difference between the performance of the explored algorithms in this dataset
- The presented algorithms can produce good recommendations, are easily applicable, and are computationally low demanding approaches, when compared to more complex algorithms

| Future work

- Explore other methods of transforming playing time into explicit ratings
- Explore other types of implicit expression of preferences, if available

Thank you!

Questions?

From Implicit Preferences to Ratings Video Games Recommendation based on Collaborative Filtering

Rosária Bunga¹, Fernando Batista^{1,2}, Ricardo Ribeiro^{1,2}

¹ Iscte - Instituto Universitário de Lisboa, Lisboa, Portugal

² INESC-ID Lisboa, Portugal

{rosaria_patricia_bunga, fernando.batista, ricardo.ribeiro}@iscte-iul.pt

Motivation and goals

Propose a videogame-oriented recommendation system

- Provide suggestions based on a **user gaming history** and the **tastes of similar users**
- Study the performance of **collaborative filtering algorithms** over Steam platform, containing a considerable amount of data
- Understand how to use implicit data to infer explicit user ratings
- Check if playing time of the users serves as an adequate implicit representation of the user's preferences

Dataset and setup

- Purchase history of Australian users of the video game *Steam* platform (Pathak et al., 2017).
- For each user, it provides the list of purchased items with a small collection of metadata related to game playing time.

Attribute	Type	Description
user_id	string	User
steam_id	integer	Steam user identification
user_url	string	User URL
item_id	integer	Game identification
item_name	string	Game title
playtime_2weeks	integer	Number of minutes played in the last two weeks
playtime_forever	integer	Total number of minutes played

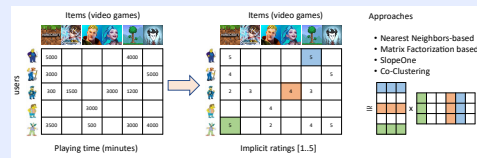
Setup

- We use a filtered dataset
 - Removed: players with less than 30 games, games that were played by less than 10 players, and games that had a playing time of zero minutes
 - **33307** unique users, **6387** unique games, and about **2.8 million** records
 - 80% for training and 20% for testing (Info: 10%, Target: 10%)
- Our experiments use the implementations of the **Surprise library**
 - allows to create and evaluate recommender systems for explicit data
- The performance of the different approaches was evaluated using common metrics used in the evaluation of recommendation systems
 - RMSE, MAE, Precision@k, Recall@k and F1@k

Approaches

From implicit preferences to ratings

- We used the **total playing time to infer the explicit ratings**, and to understand the preferences of the users. It is used to quantify the relevance of an item to a specific user, on the assumption that **if a user plays a game for a long time, he likes that game**.
- For converting implicit ratings into explicit ratings, we grouped the data by user and playing time, and then we grouped the playing times into five bins. Then for each group we computed its rank, using a scale ranging from 1 to 5.



Recommendation Approaches

- Nearest Neighbors-based Algorithms
- Algorithms based on Matrix Factorization
- SlopeOne
- Co-Clustering

Conclusions and future work

- We have tested 7 recommendation systems based on collaborative filtering for the video game domain
- Before training the algorithms, the **users' total playing time** was converted into an explicit rating

Conclusions

- SlopeOne and SVD++ were the best performing recommendation algorithms. Considering computational cost, SlopeOne is the best
- Considering the dataset under study, there is not a significant difference between the performance of the explored algorithms
- It is possible to use playtime to infer explicit ratings for making recommendations in the video game domain and specifically on the Steam platform
- The presented algorithms can produce good recommendations, are easily applicable, and are computationally low demanding approaches, when compared to more complex algorithms.

Future work

- Explore other methods of transforming playing time into explicit ratings; Explore other types of implicit expression of preferences, if available.

Results

– The algorithms show similar performances.

	RMSE	MAE	precision			recall			F1	
			@5	@10	@20	@5	@10	@20	@5	@10
KNNBasic	1.2270	1.0159	0.7374	0.7144	0.7064	0.2866	0.3718	0.3954	0.4128	0.4891
KNNWithMeans	1.2483	1.0356	0.7293	0.7043	0.6955	0.2767	0.3595	0.3841	0.4012	0.4760
SVD	1.2386	1.0049	0.7619	0.7162	0.7003	0.3367	0.4609	0.5028	0.4670	0.5608
SVD++	1.2179	0.9727	0.7757	0.7309	0.7145	0.3397	0.4640	0.5095	0.4725	0.5677
NMF	1.2229	1.0011	0.7525	0.7073	0.6920	0.3275	0.4470	0.4882	0.4564	0.5478
SlopeOne	1.1977	0.9831	0.7840	0.7336	0.7183	0.3391	0.4535	0.4920	0.4735	0.5605
Co-Clustering	1.2354	1.0212	0.7713	0.7288	0.7150	0.3174	0.4229	0.4558	0.4498	0.5352

- Overall, **SlopeOne** and **SVD++** achieved the best results for all metrics.
- Considering the RMSE and MAE, SlopeOne and SVD++ exhibited the best results
- Concerning precision, SlopeOne achieved the best result
- Concerning recall, SVD++ performed better, not much different from SVD
- KNNWithMeans had the worst results
- The distribution of the predicted ratings on the test set is noticeably different for all algorithms, not reflecting the actual distribution of implicit ratings.
- Concerning the average training time, SlopeOne takes about 24 seconds while SVD++ takes about 4904 seconds (~200x), which corresponds to a significant difference. Concerning the testing time, SlopeOne takes about 77 seconds while SVD++ takes 110 seconds, being about 42% slower.