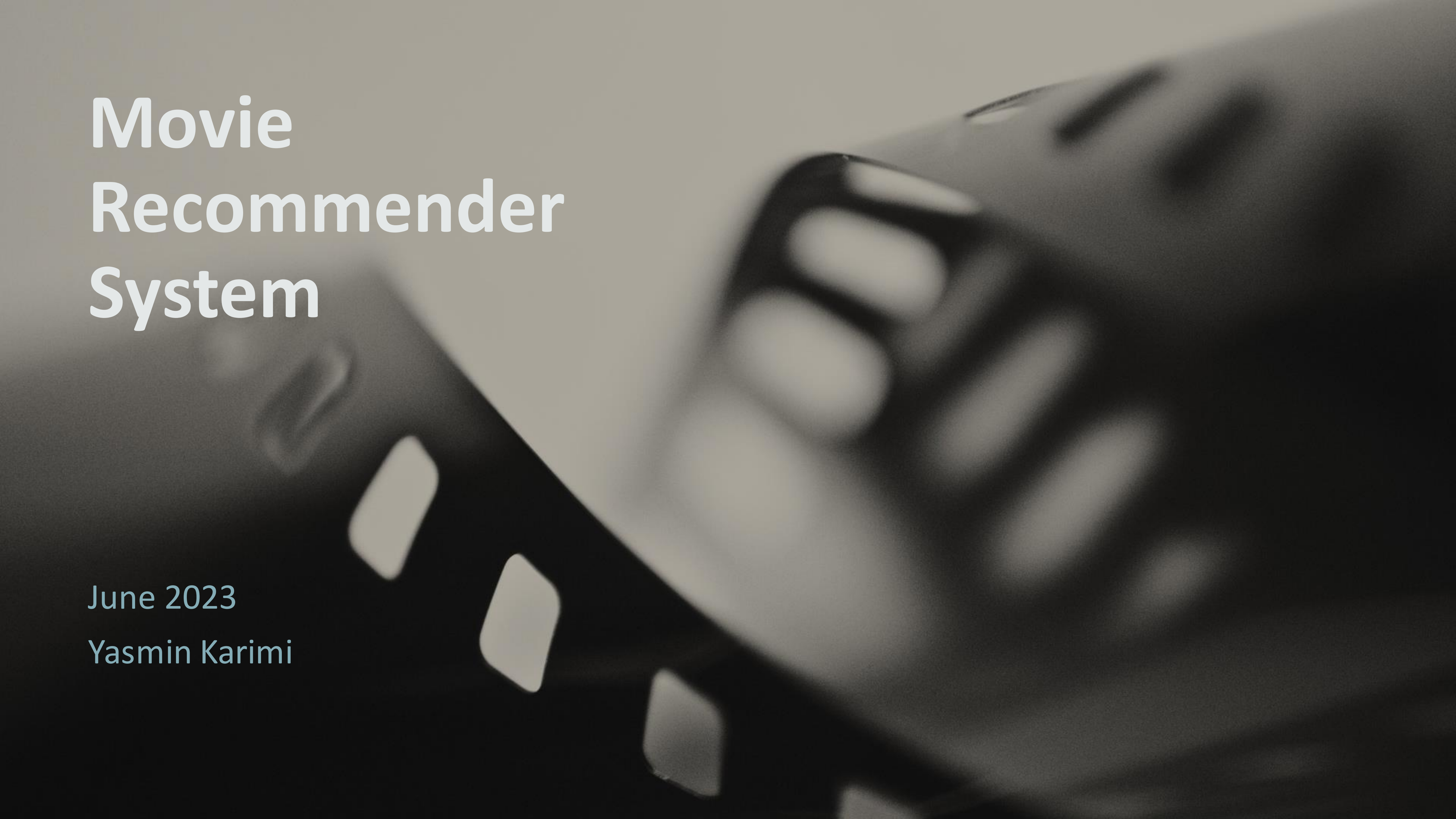
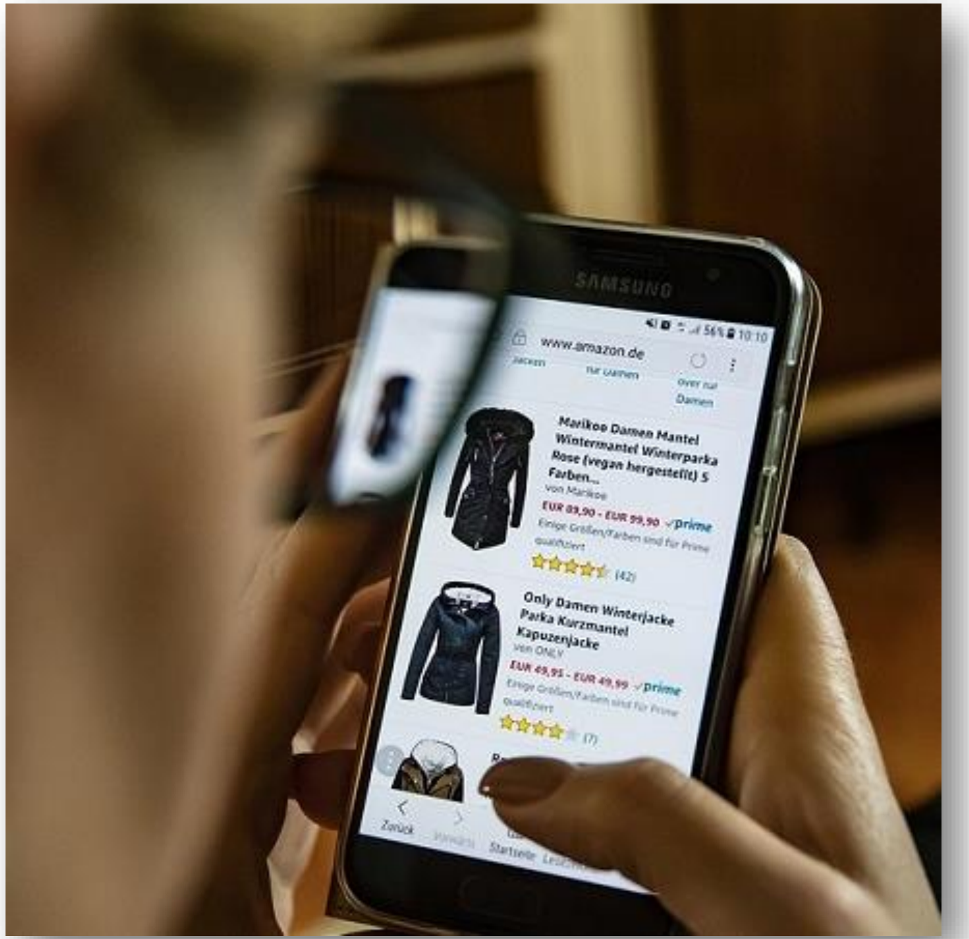


Movie Recommender System

June 2023

Yasmin Karimi



Recommender systems are designed to recommend products to the user based on many different factors. These systems predict the most likely product that the users are most likely to purchase and are interested in.

- A customer is willing to purchase at a shop where they're getting maximum help in scouting the right product
- . They're also much more likely to return to such a shop in the future.
- Using recommender systems in many businesses can help with increasing: revenue, Customer satisfaction, personalization, etc.
- It's said that Amazon's AI-powered recommendation engine is responsible for over one-third of its sales, making it one of the most valuable AI systems on the planet.

Data used for this project

'Movie lens' full dataset.

Original Data :

- Good amount of reviews (~27 millions of reviews)
- Approximately 45,000 movies and 24 features (information of the movies)

Movie Id	Int
title	String
Overview	String
Genre	List

User ID	Int
Movie ID	Int
Rating	Float

movielens

kaggle

How it works?

The system is constantly collecting data from you. (your actions, purchases, etc)

For each user profile it has a browsing history, it knows what are your likes and dislikes through the previous interactions you had.

Why? For better experience and to make the process of purchasing an item, or in this case to choose a movie to watch easier, it will recommend you a new item.

This recommendation is based on your previous interactions with two main approaches:

- User-based filtering
- Item-based Collaborative filtering

Item-based collaborative filtering was developed by Amazon.

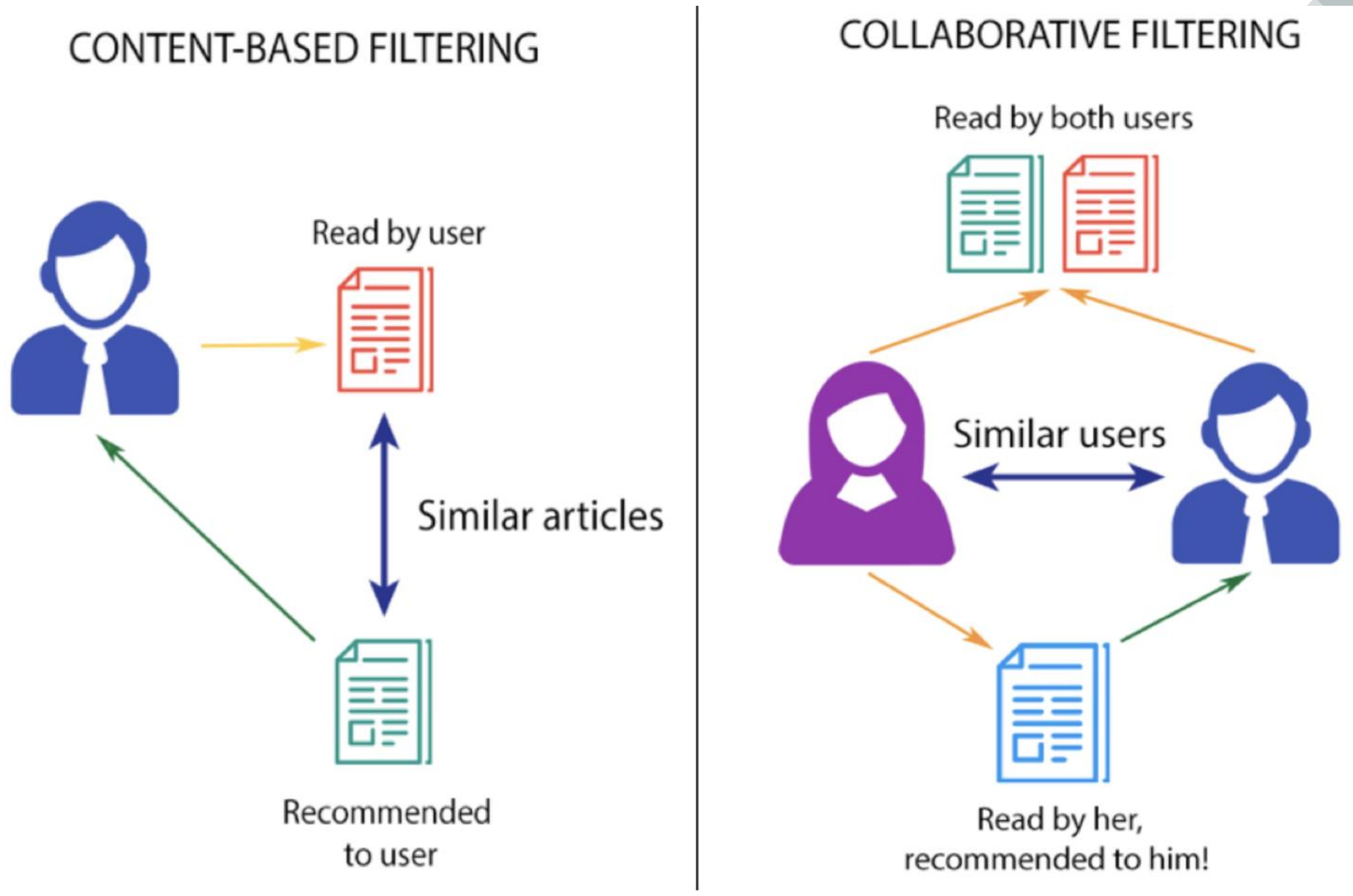
In a system where there are more users than items, item-based filtering is faster and more stable than user-based.

Item-based: performs poorly for datasets with browsing or entertainment related items such as 'Movie Lens' (This project).

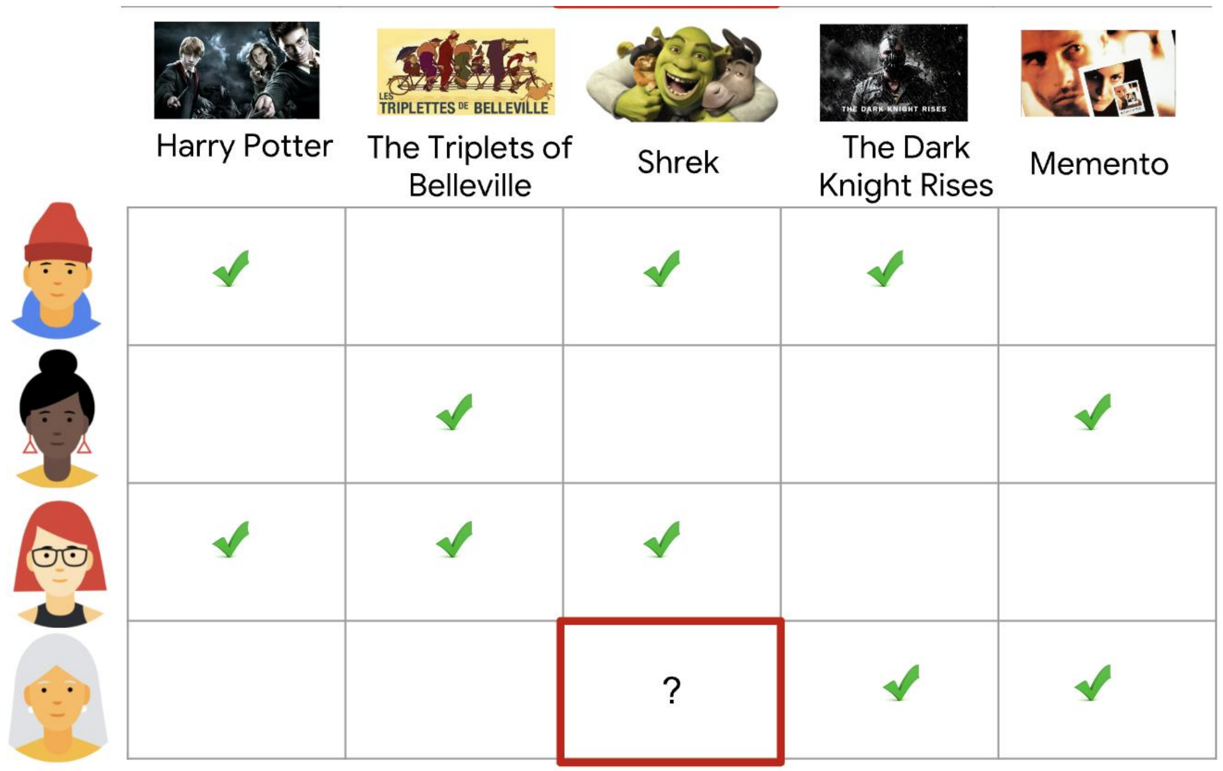
To see better results:

1.matrix factorization techniques

2. hybrid recommenders


CONTENT-BASED FILTERING
Read by user
Similar articles
Recommended to user
COLLABORATIVE FILTERING
Read by both users
Similar users
Read by her, recommended to him!

Collaborative filtering



	Harry Potter	The Triplets of Belleville	Shrek	The Dark Knight Rises	Memento
	✓		✓	✓	
		✓			✓
	✓	✓	✓		
			?	✓	✓

☒ • Cold start problem

☑ • The model can help users discover new interests

Algorithm used in this project:
SVD (Singular Value Decomposition)
SVD came into the limelight when matrix factorization was seen performing well in the Netflix prize competition.

Results

Collaborative filtering: Accurate~ 96% on training set, no test set

Content-based filtering: Accurate

User-based filtering: Subjective results / not necessarily correct information

Potential next steps / Improvements

- **SoftMax DNN for Recommendation**

One possible DNN model is SoftMax, which treats the problem as a multiclass prediction problem in which:

- The input is the user query.
- The output is a probability vector:

with size equal to the number of items in the corpus, representing the probability to interact with each item; for example, the probability to click on or watch a video.

Input

The input to a DNN can include:

- dense features (for example, watch time and time since last watch)
- sparse features (for example, watch history and country)

Unlike the matrix factorization approach, you can add side features such as age or country. We'll denote the input vector by x.

Output

The new model predicts one value for each pair from the last layer.

	Matrix Factorization	Softmax DNN
Query features	🚫 Not easy to include.	✓ Can be included.
Cold start	🚫 Does not easily handle out-of vocab queries or items. Some heuristics can be used (for example, for a new query, average embeddings of similar queries).	✓ Easily handles new queries.
Folding	✓ Folding can be easily reduced by adjusting the unobserved weight in WALS.	🚫 Prone to folding. Need to use techniques such as negative sampling or gravity.
Training scalability	✓ Easily scalable to very large corpora (perhaps hundreds of millions items or more), but only if the input matrix is sparse.	🚫 Harder to scale to very large corpora. Some techniques can be used, such as hashing, negative sampling, etc.
Serving scalability	✓ Embeddings U, V are static, and a set of candidates can be pre-computed and stored.	✓ Item embeddings V are static and can be stored. 🚫 The query embedding usually needs to be computed at query time, making the model more expensive to serve.

Potential next steps / Improvements

Use case of the project

- The application of Recommender systems are broad. they can be used in: Media, retail, banking, etc.
- For this case it can potentially be applied in a website for recommending movies or tv-shows to people.
- Consider Netflix, for example. About 75% of what users watch on Netflix is a result of its product recommendation algorithm.

Thank you!