

RECOMMENDER SYSTEMS BASED ON COLLABORATIVE FILTERING

Faeze Saeedi Nejad | Spring 2021

Topics

- Concepts
 - Understanding you
 - Top-N recommender
 - Installations
 - SurpriseLib
 - Evaluation
 - Errors
 - Metrics
 - Recommender engine
 - AlgoBase
 - Similarity metrics
 - User-based collaborative filtering
 - Item-based collaborative filtering
 - Conclusion
 - References
-

Concepts

- Recommender systems make it easy for user to find what they want using their previous information.
- Before recommending, recommender system should understand users and their interests to use that data for recommendations.

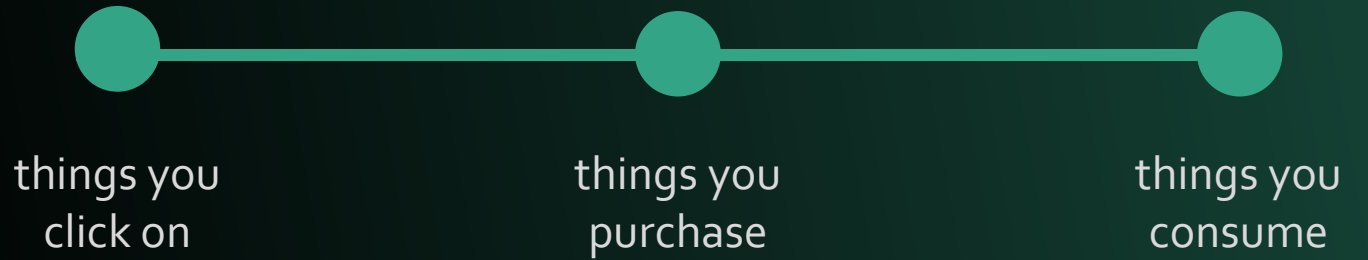


Concepts

- Understanding you explicitly
 - Sparsity
 - Different standards

Concepts

- Understanding you implicitly
 - Like: lots of Amazon purchase data




Explicit movie ratings using MovieLens data

Concepts


- Top-N recommender

Music View All & Manage


Page 1 of 20



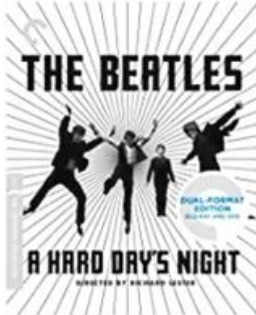
Star Trek: Discovery (Original Series Soundtrack) [Chapter 2]
Jeff Russo
★★★★☆ 1
\$11.29 ✓prime




Solo: A Star Wars Story (Original Motion Picture Soundtrack)
John Powell
★★★★☆ 17
\$11.88 ✓prime




The Beatles: Help! [Blu-ray]
John Lennon
★★★★☆ 768
\$22.49 ✓prime



A Hard Day's Night (Criterion Collection) (Blu-ray + DVD)
John Lennon
★★★★☆ 455
\$29.10 ✓prime



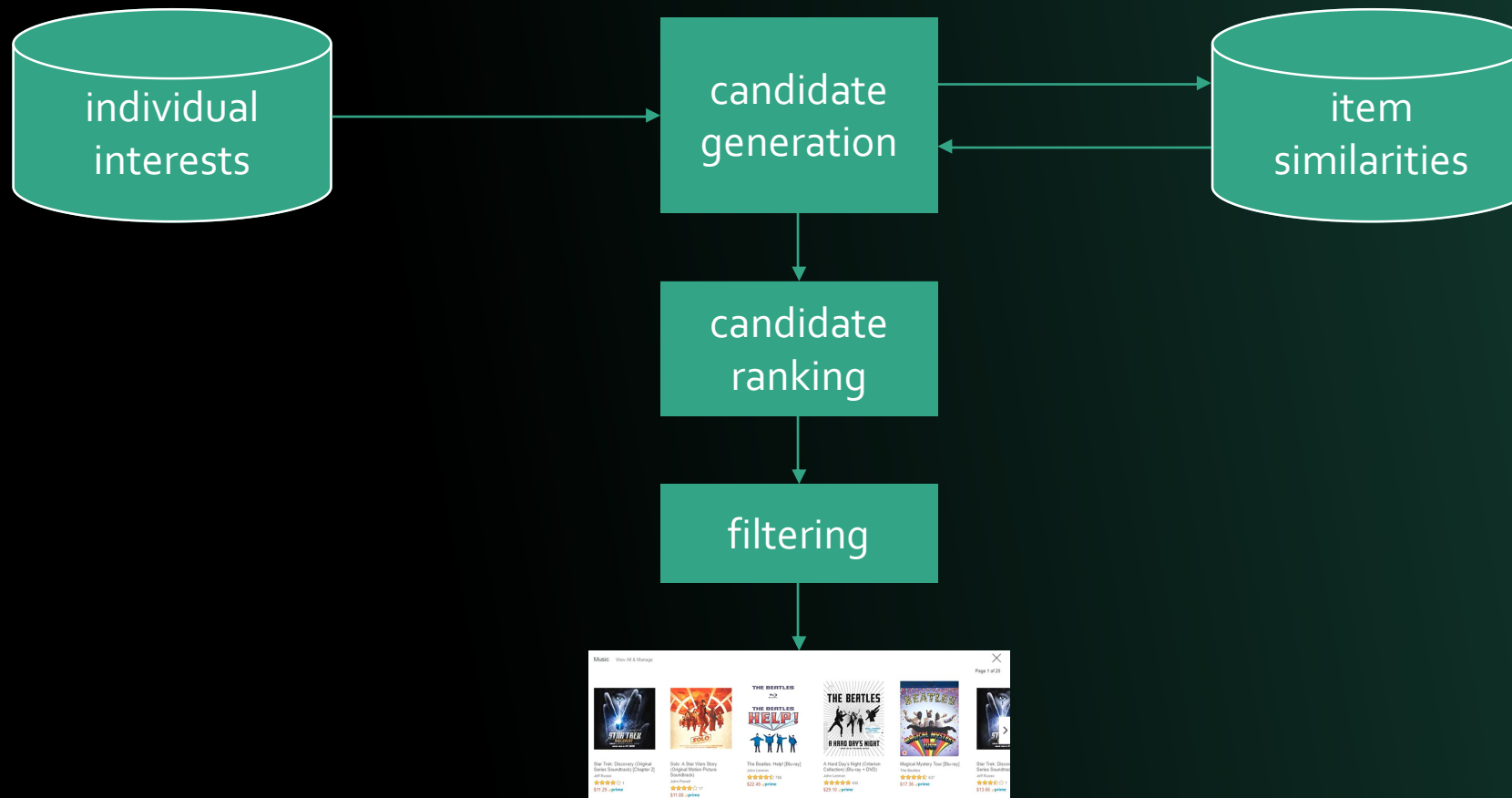
Magical Mystery Tour [Blu-ray]
The Beatles
★★★★☆ 637
\$17.36 ✓prime



Star Trek: Discovery Series Soundtrack
Jeff Russo
★★★★☆ 7
\$13.66 ✓prime

Concepts

– How it works?



Installation

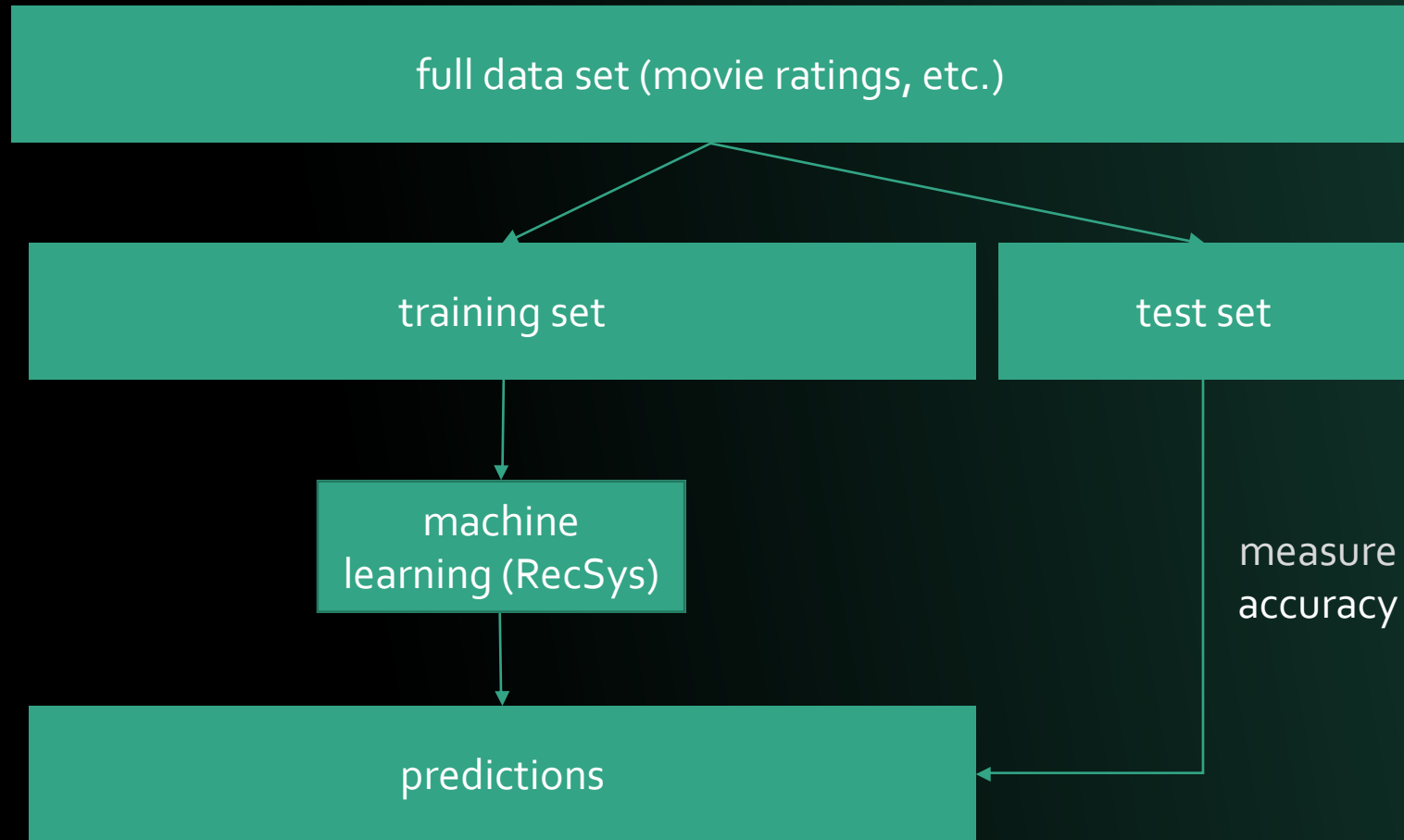
- SurpriseLib
- SVD

Evaluation

- A big part of why recommender systems are as much art as they are science is that it's difficult to measure how good they are!

Evaluation

- Methodology for testing offline



Evaluation

- How accurately you can predict how users rated movies they've already seen and provided a rating for
- That's not the point!
- Recommending new things is the point, things that users find interesting

Evaluation

- More ways to evaluate

Evaluation

- Mean absolute error (MAE)

$$\frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

predicted ratings

5
4
5
1

predicted ratings

3
1
4
1

error

2
3
1
0

$$\text{MAE} = (2+3+1+0)/4 = 1.5$$

Evaluation

- Root mean square error (RMSE)
 - More fancy way to measure accuracy
 - Penalization

$$\sqrt{\frac{\sum_{i=1}^n (y_i - x_i)^2}{n}}$$

predicted ratings	predicted ratings	error ²
5	3	4
4	1	9
5	4	1
1	1	0

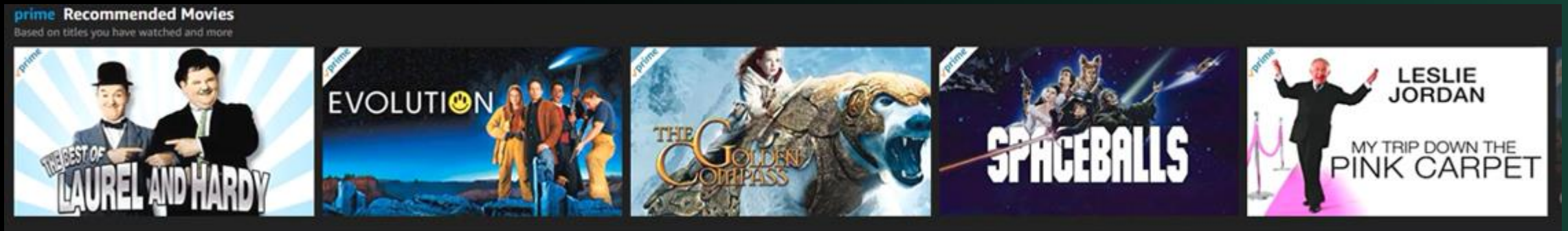
$$\text{MAE} = \sqrt{(4+9+1+0) / 4} = 1.87$$

Evaluation

- But that's not what the user wants, they don't care if your system is good/bad at predicting the ratings, users want to see top-N recommendations for them
- That's why Netflix didn't use the algorithm from BellKor team in 2009 (low RMSE score)

Evaluation

- Metrics that are more focused on top-N recommenders



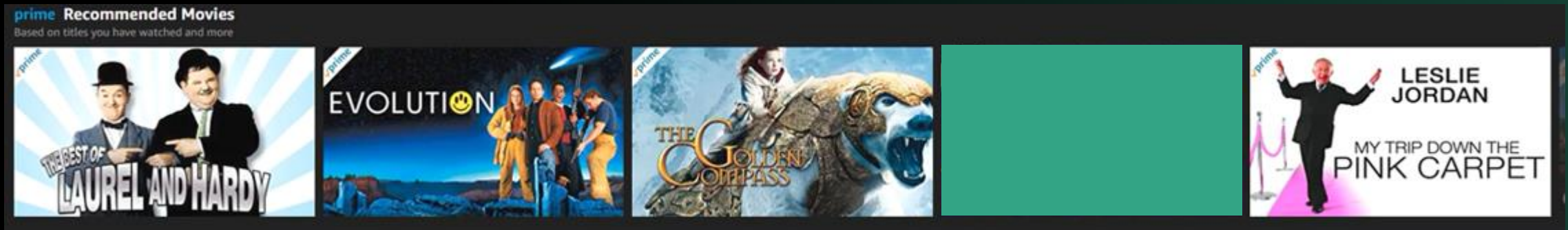
Evaluation

- Hit rate
- Using test set, you create top-N recommendation for each user, if they've already seen it and liked it, it's considered as a hit

$$\frac{hits}{users}$$

Evaluation

- Easy to understand but hard to measure!
- Leave-one-out cross validation
- Measuring the ability to recommend the left-out movie in test phase



Evaluation

- Accuracy isn't the only thing that matters when it comes to evaluating recommender systems

Evaluation

- Coverage
- Bigger dataset = better coverage score

% of <user, item> pairs that can be provided

Evaluation

- Diversity
- How broad of variety items the recommender system is putting in front of people
- Low diversity: recommending just the next book in Harry Potter series

$$(1 - S)$$

S = avg similarity between recommendation pairs

Evaluation

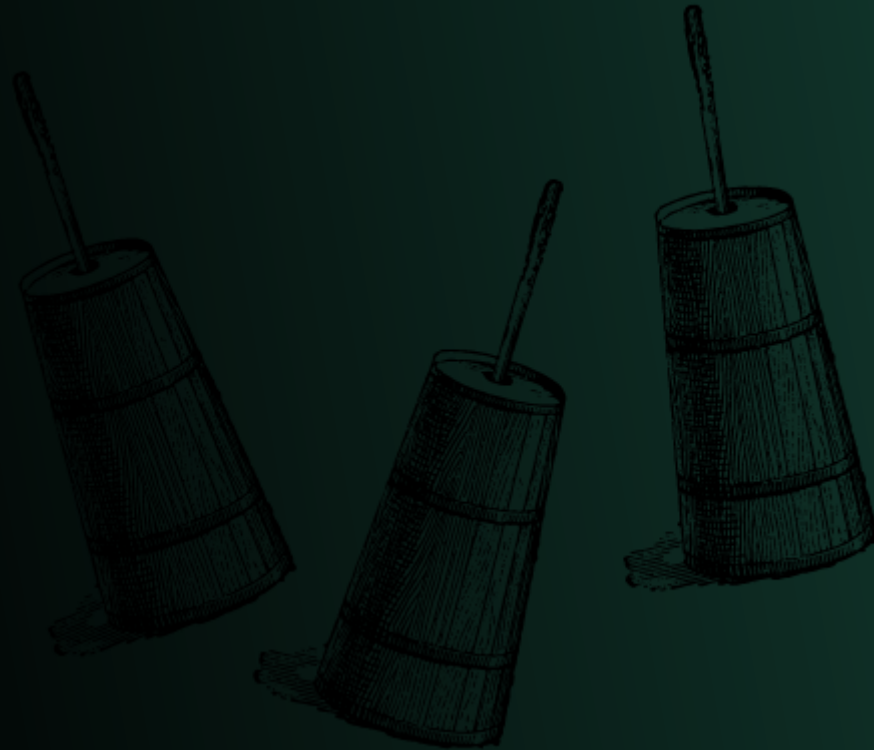
- Remember that high diversity score isn't good, you can achieve it by just recommending random stuff
- Look at diversity along side the metrics that measure quality, and there should be a balance

Evaluation

- Novelty
- Popularity rank of recommended items
- But user-trust is more important, so making a trade-off between new items and familiar items to user, is important
- New items -> discovering / familiar items -> user-trust

Evaluation

- Churn
- How often do recommendations change?
- System's sensitivity to user's behavior



Evaluation

- Responsiveness
- How quickly does new user's behavior influence your recommendations?
- Strike a balance between simplicity and responsiveness

What's important?

Evaluation

- Online A/B testing
- Tuning the recommender system using real customers
- understanding by user's behavior or measure perceived quality
- Users reactions to the recommender system is way more important than the metrics we discussed

Evaluation

- It takes too long to run, why? cause it needs to predict ratings
- Average 0.7 error predicting rating
- Diversity and novelty are high, don't forget user-trust

```
Evaluating accuracy of model...
RMSE: 0.9033701087151801
MAE: 0.6977882196132263

Evaluating top-10 recommendations...
Computing recommendations with leave-one-out...
Predict ratings for left-out set...
Predict all missing ratings...
Compute top 10 recs per user...

Hit Rate: 0.029806259314456036

Computing complete recommendations, no hold outs...

User coverage: 0.9552906110283159
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.

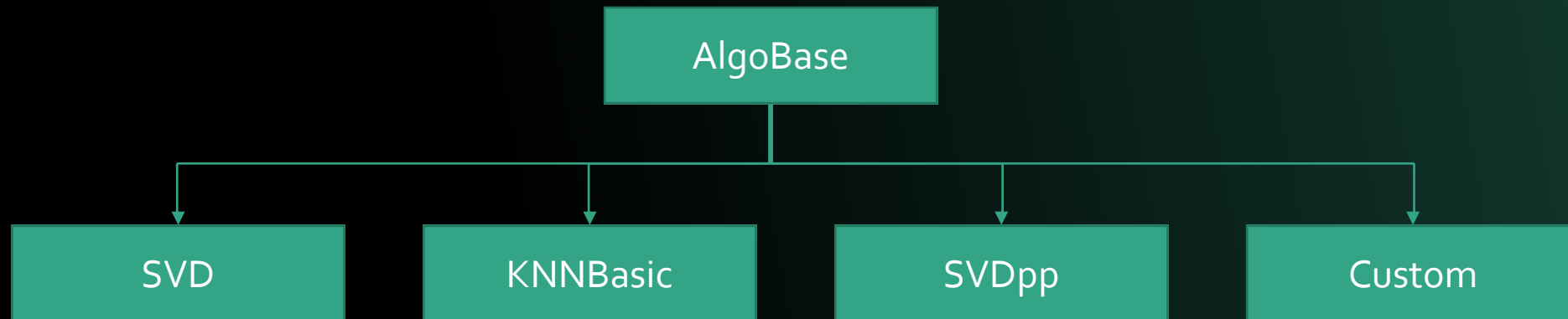
Diversity: 0.9665208258150911

Novelty (average popularity rank): 491.5767777960256

In [2]:
```

Recommender engine

- But we need a framework
- SurpriseLib algorithm base class



Recommender engine

- SVD vs random algorithm
- Comparison using base class functions

Algorithm	RMSE	MAE	HR	Coverage	Diversity	Novelty
SVD	0.9034	0.6978	0.0298	0.9553	0.0445	491.5768
Random	1.4355	1.1468	0.0119	1.0000	0.0694	531.7888

Legend:

RMSE: Root Mean Squared Error. Lower values mean better accuracy.

MAE: Mean Absolute Error. Lower values mean better accuracy.

HR: Hit Rate; how often we are able to recommend a left-out rating. Higher is better.

Coverage: Ratio of users for whom recommendations above a certain threshold exist. Higher is better.

Diversity: $1-S$, where S is the average similarity score between every possible pair of recommendations for a given user. Higher means more diverse.

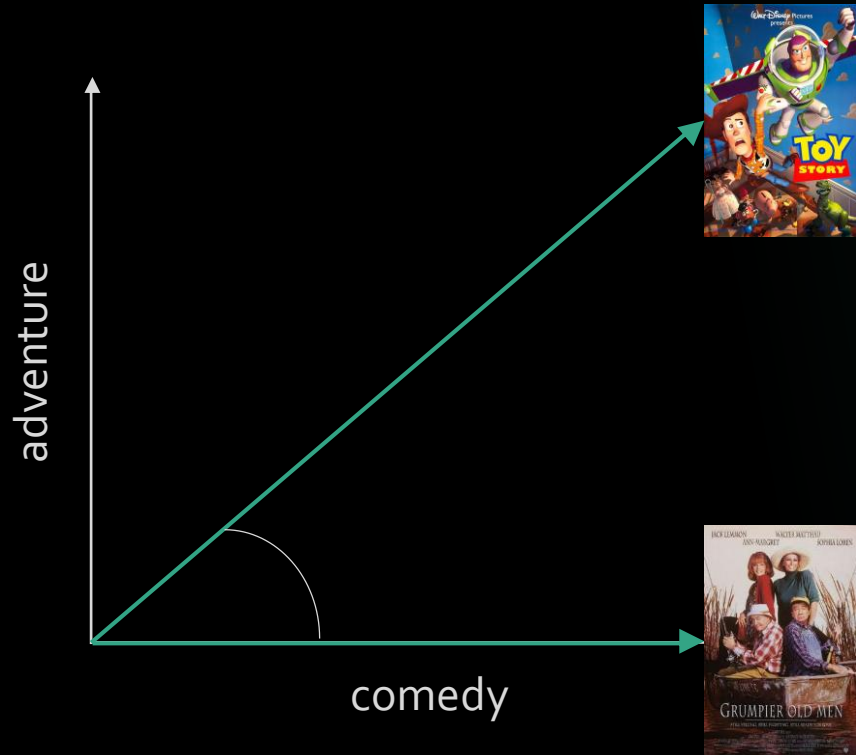
Novelty: Average popularity rank of recommended items. Higher means more novel.

In [2]:

Recommender engine

- Content-based filtering
- Many ways to measure similarity score
- Cosine similarity is the most popular

Recommender engine

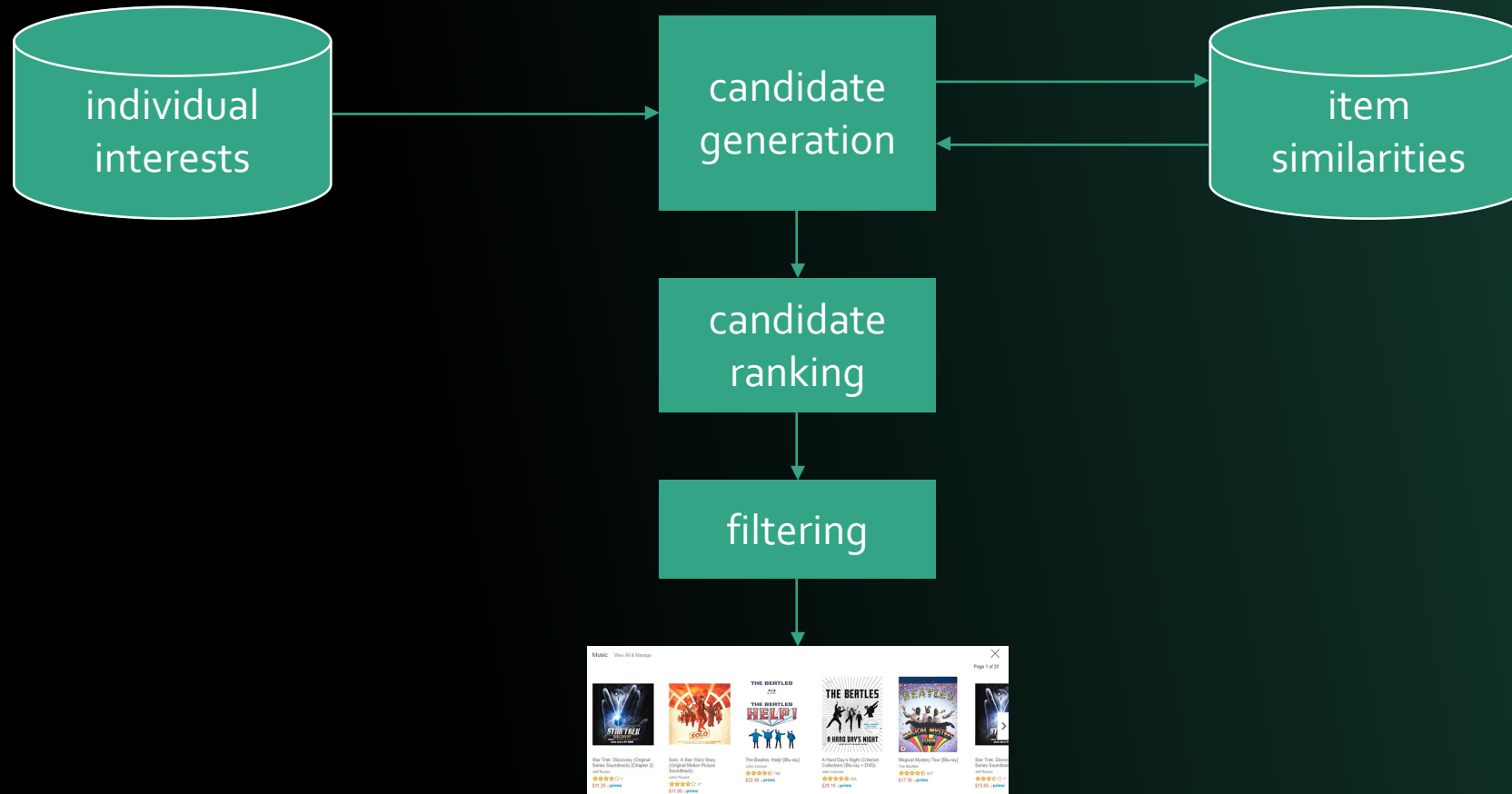


- Cosine similarity
- 45° isn't a useful similarity metric, we want something between 0 and 1
- The cosine of the angle does just that
- 0.7 is the cosine similarity score
- multi dimensional

$$\text{CosSim}(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}}$$

Recommender engine

- Collaborative filtering
- Finding similar users/items is the challenge

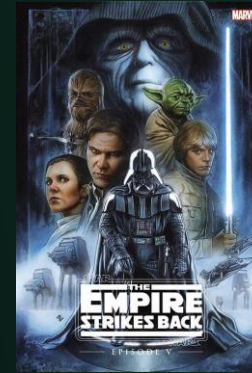


Recommender engine

- In cosine similarity, consider user's behavior as dimensions instead of genres
- Sparsity is the challenge while using cosine similarity
- Adjusted cosine is what we use

Recommender engine

- User-based collaborative filtering
- Finding similar users based on ratings history



Recommender engine

- Table of all the ratings from everyone in our system
- 2-d array and then a table for cosine similarity

	Indiana Jones	Star Wars	Empire Strikes Back	Incredibles	Casablanca
Bob	4	5			
Ted					1
Ann		5	5	5	



	Bob	Ted	Ann
Bob	1	0	1
Ted	0	1	0
Ann	1	0	1

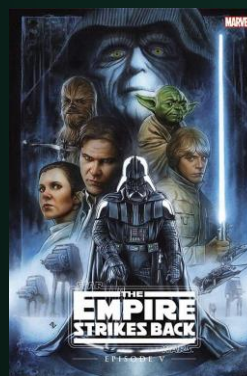
Bob's neighbors: Ann: 1.0, Ted: 0

Recommender engine

- So we consider Ann in Bob's top-N similar users
- In a less sparse data, there would be a better top-N users other than just Ann
- Recommendation candidates



1.0



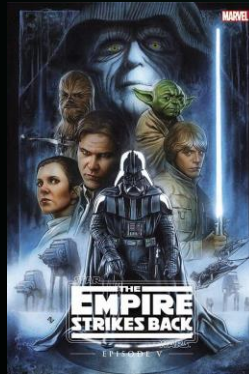
1.0

Recommender engine

- Sort the candidates then filter them



1.0



1.0



Recommender engine

- Hands on
- No need to split test/train set cause we're not measuring accuracy, we're just making top-N recommendations
- It's quick

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Inception (2010) 3.3
Star Wars: Episode V - The Empire Strikes Back (1980) 2.4
Bourne Identity, The (1988) 2.0
Crouching Tiger, Hidden Dragon (Wo hu cang long) (2000) 2.0
Dark Knight, The (2008) 2.0
Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il) (1966) 1.9
Departed, The (2006) 1.9
Dark Knight Rises, The (2012) 1.9
Back to the Future (1985) 1.9
Gravity (2013) 1.8
Fight Club (1999) 1.8
```

```
In [3]:
```

Recommender engine

- Item-based collaborative filtering
- Things, not people cause things are more permanent nature than people
- Far fewer things, smaller matrix, faster computing
- User-experience

Recommender engine

- Just flip the users and items
- Measure similarity between items based on ratings

	Bob	Ted	Ann
Indiana Jones	4		
Star Wars	5		5
Empire Strikes Back			5
Incredibles			5
Casablanca		1	



	Indiana Jones	Star Wars	Empire Strikes Back	Incredibles	Casablanca
Indiana Jones	1	1	0	0	0
Star Wars	1	1	1	1	0
Empire Strikes Back	1	1	1	1	0
Incredibles	1	1	1	1	0
Casablanca	0	0	0	0	1

Recommender engine

- Hands on
- Similar cause the general approach is the same
- Focus the relationship between items
- Amazon uses this

```
Computing the cosine similarity matrix...  
Done computing similarity matrix.  
Computing the cosine similarity matrix...  
Done computing similarity matrix.  
James Dean Story, The (1957) 10.0  
Get Real (1998) 9.987241120712646  
Kiss of Death (1995) 9.966881877751941  
Set It Off (1996) 9.963732215657119  
How Green Was My Valley (1941) 9.943984081065269  
Amos & Andrew (1993) 9.93973694500253  
My Crazy Life (Mi vida loca) (1993) 9.938290487546041  
Grace of My Heart (1996) 9.926255896645218  
Fanny and Alexander (Fanny och Alexander) (1982) 9.925699671455906  
Wild Reeds (Les roseaux sauvages) (1994) 9.916226404418774  
Edge of Seventeen (1998) 9.913028764691676
```

```
In [4]: |
```

Recommender engine

– Hit rate

```
Computing movie popularity ranks so we can measure novelty later...  
Estimating biases using als...  
Computing the cosine similarity matrix...  
Done computing similarity matrix.  
Computing the cosine similarity matrix...  
Done computing similarity matrix.  
Computing the cosine similarity matrix...  
Done computing similarity matrix.  
HR 0.05514157973174367
```

```
In [6]:
```

Conclusion

- Evaluating is important
- SurpriseLib made implementation easy
- More data to work with = better recommender system

Github link:

<https://github.com/fzsdi/collaborative-filtering>

References

- Building Recommender Systems with Machine Learning and AI (Frank Kane)
- Build a Recommendation Engine With Collaborative Filtering (Abhinav Ajitsaria)
- How to Build Simple Recommender Systems in Python (Bryan Tan)
- Other references are in the thesis file

Thank you

