Computational Communication Science 2
Week 6 - Lecture
»Recommender systems«

Anne Kroon

a.c.kroon@uva.nl, @annekroon

May, 2022

Digital Society Minor, University of Amsterdam

Today

Recommender Systems

Knowledge-based Recommender Systems

Content-based Recommender Systems

Collaborative Recommender Systems

Group Assignment

Recommender Systems


You
Tube

Recommender Systems in Communication Science

New research questions

1. Political communication and journalism. E.g., to craft personalized news diets. This may have, however, consequences for the diversity of news diets and for democracy (Locherbach & Trilling, 2018; Möller et al., 2018)
2. Organizational and corporate communication. E.g., applications in the field of hiring and recruitment.
3. Persuasive communication. E.g., in the health domain–recommendation algorithms for tailored health interventions (Kim et al., 2019)
4. Entertainment communication. E.g., movie recommenders

Recommender Systems

Two central problems within Recommender Systems

1. Predicting problem. Typical problem involves a lot of missing data (e.g., user only rated a small subset of movies/news articles/ etc.) How can we deal with missing data?
2. Ranking problem. Given n items, can we identify the top k items to recommend?

These problems are not isolated; rather, they are connected.

Recommender Systems

How do recommender systems learn?

1. Explicit user preferences. Ratings or responses
2. Implicit user preferences. E.g., clicks or viewing time
3. Content. E.g., based on text similarity techniques

Recommender Systems

Types of recommender systems (Locherbach & Trilling, 2018; Möller et al., 2018; Wieland et al., 2021)

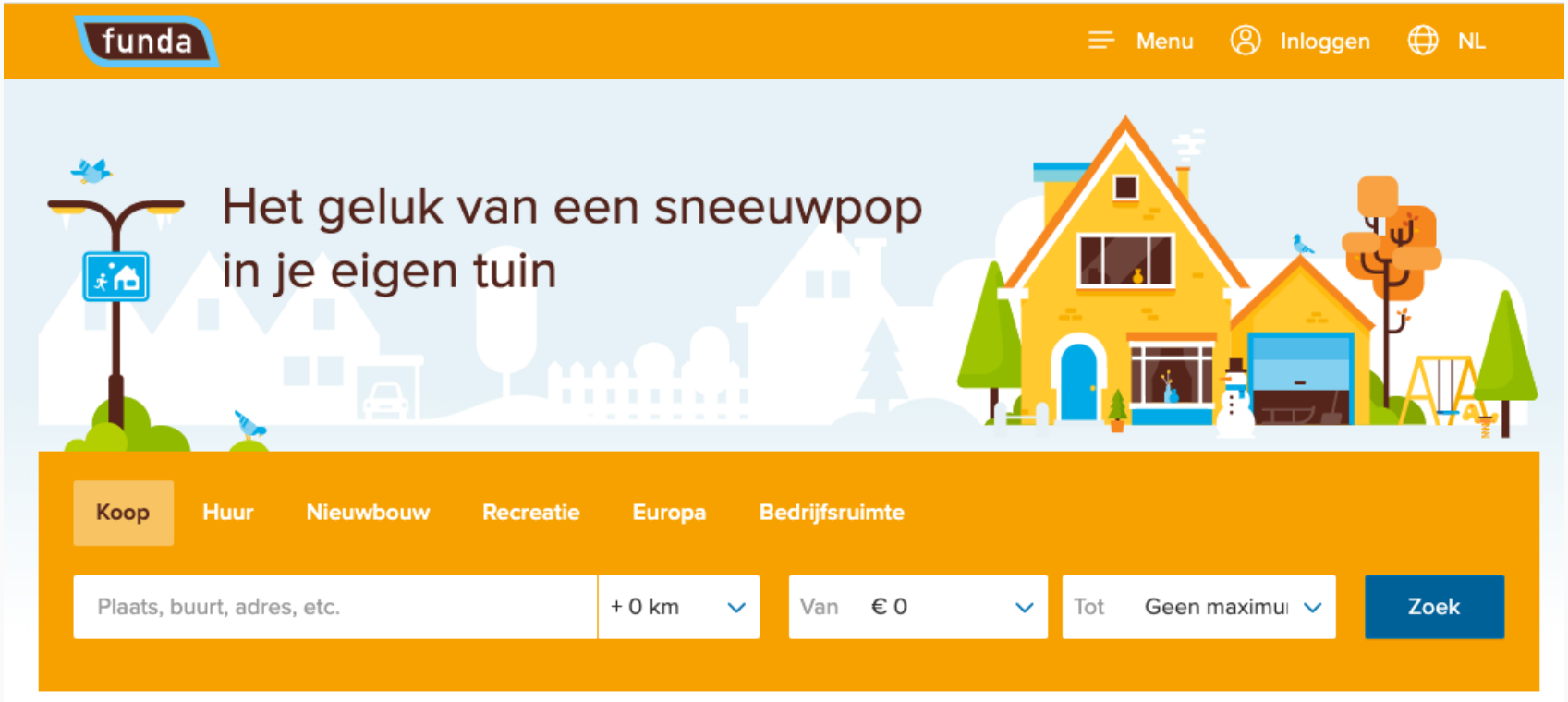
1. 'Basic' knowledge-base recommender systems
2. Content-based recommender systems
3. Collaborative recommenders

Knowledge-based RecSys

Knowledge-based recommender system

When to use?

- To overcome the **cold start problem**; when we do not have ratings of individual users.
- Simple model. It does not rely on user's explicit or implicit ratings, but on specific queries.
- Typical use case: Real-estate. Buying a house is, for most families, a rare/ single event.


funda
Menu
Inloggen
NL
Het geluk van een sneeuwpop
in je eigen tuin
Koop
Huur
Nieuwbouw
Recreatie
Europa
Bedrijfsruimte
Plaats, buurt, adres, etc.
+ 0 km
Van € 0
Tot Geen maximu
Zoek

Use case: ImDb database

	genres	title	tagline	release_date	vote_average	vote_count
0	[action, adventure, fantasy, science fiction]	Avatar	Enter the World of Pandora.	2009-12-10	7.2	11800
1	[adventure, fantasy, action]	Pirates of the Caribbean: At World's End	At the end of the world, the adventure begins.	2007-05-19	6.9	4500
2	[action, adventure, crime]	Spectre	A Plan No One Escapes	2015-10-26	6.3	4466
3	[action, crime, drama, thriller]	The Dark Knight Rises	The Legend Ends	2012-07-16	7.6	9106
4	[action, adventure, science fiction]	John Carter	Lost in our world, found in another.	2012-03-07	6.1	2124

What are relevant variables to use in a knowledge-based recommender system?

Knowledge-based recommender system

How can we work with user input without a front-end (such as the website of funda? → enter python's native input() function.

```
print("What is your favorite movie genre?")
genre = input()
```

```
1 what is your favorite movie genre?
2 [...]
```

Improving knowledge-based recommender system

When to use?

- It is important to think about ways to make the recommendation relevant for individuals
- Do you have more information in your db that make your top-listed recommendations as relevant as possible?

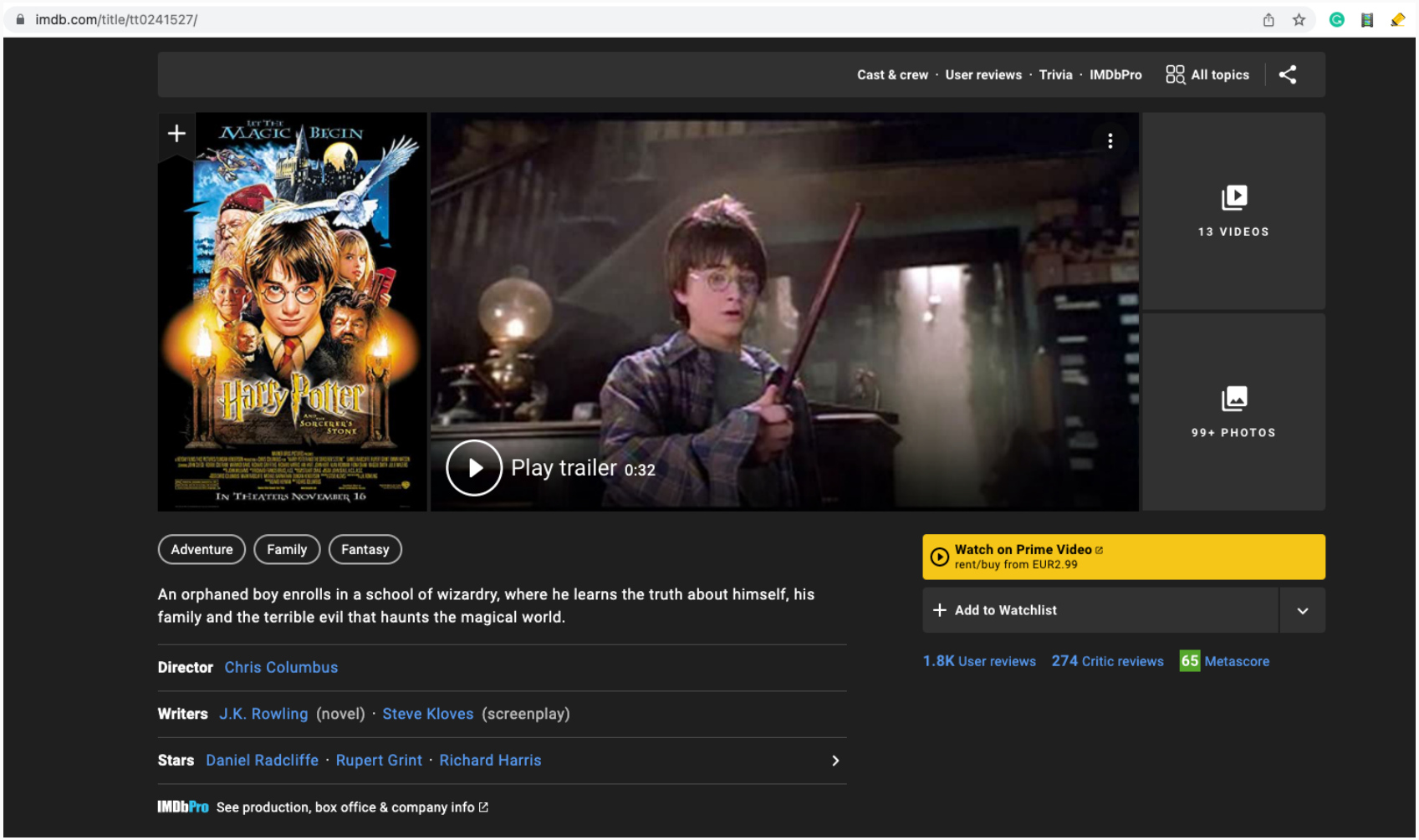
```
recommend_movies = movies.sort_values('vote_average', ascending=False)
```

Content-based RecSys

Content-based systems

- Recommends items based on user's profiles.
- Profiles are based on e.g., ratings, and represents user's tasts/ preferences.
    - For example, how often a user has clicked on, or liked, a movie.
- Recommendation is based on similarity beween items in the content.
    - Content is here: e.g., genre, tags, plot, authors, directors, location, etc.

Example of a content-based recsys



Example of a content-based recsys

More like this

★ 7.5 ☆

Harry Potter and the Chamber of Secrets

Watch options

★ 7.9 ☆

Harry Potter and the Prisoner of Azkaban

Watch options

★ 7.7 ☆

Harry Potter and the Goblet of Fire

Watch options

★ 7.5 ☆

Harry Potter and the Order of the Phoenix

Watch options

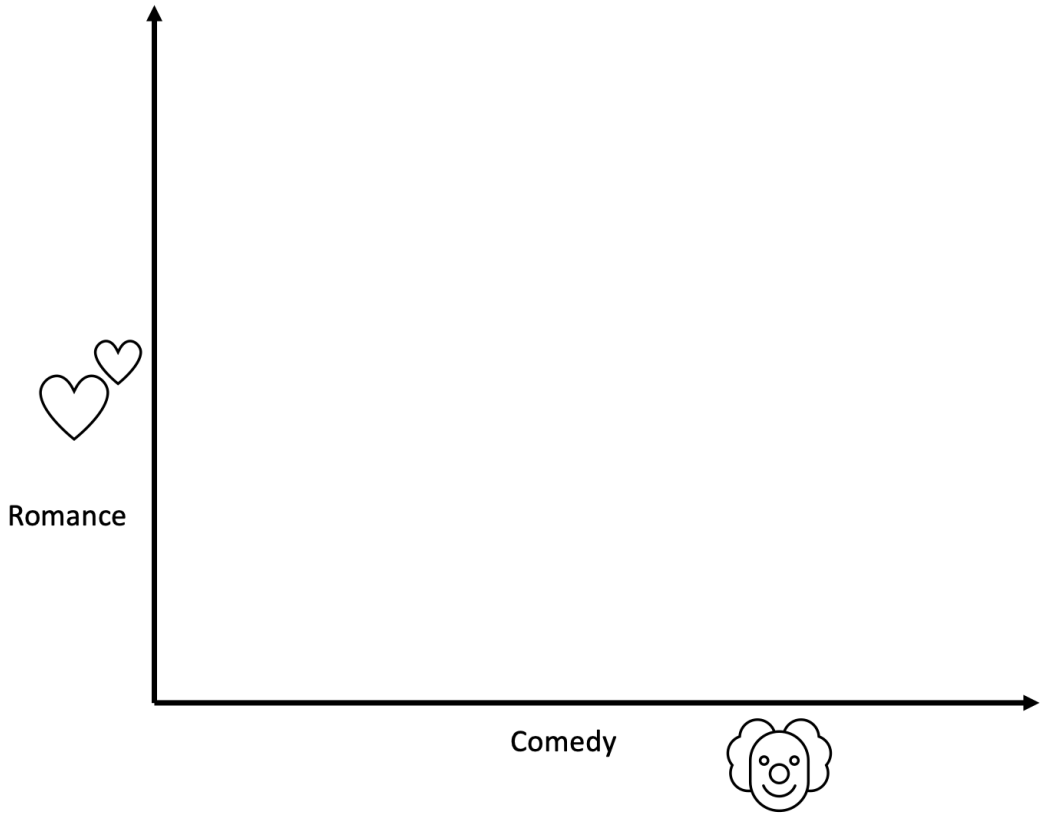
Use case: ImDb database

Let's have a look at our use case again.

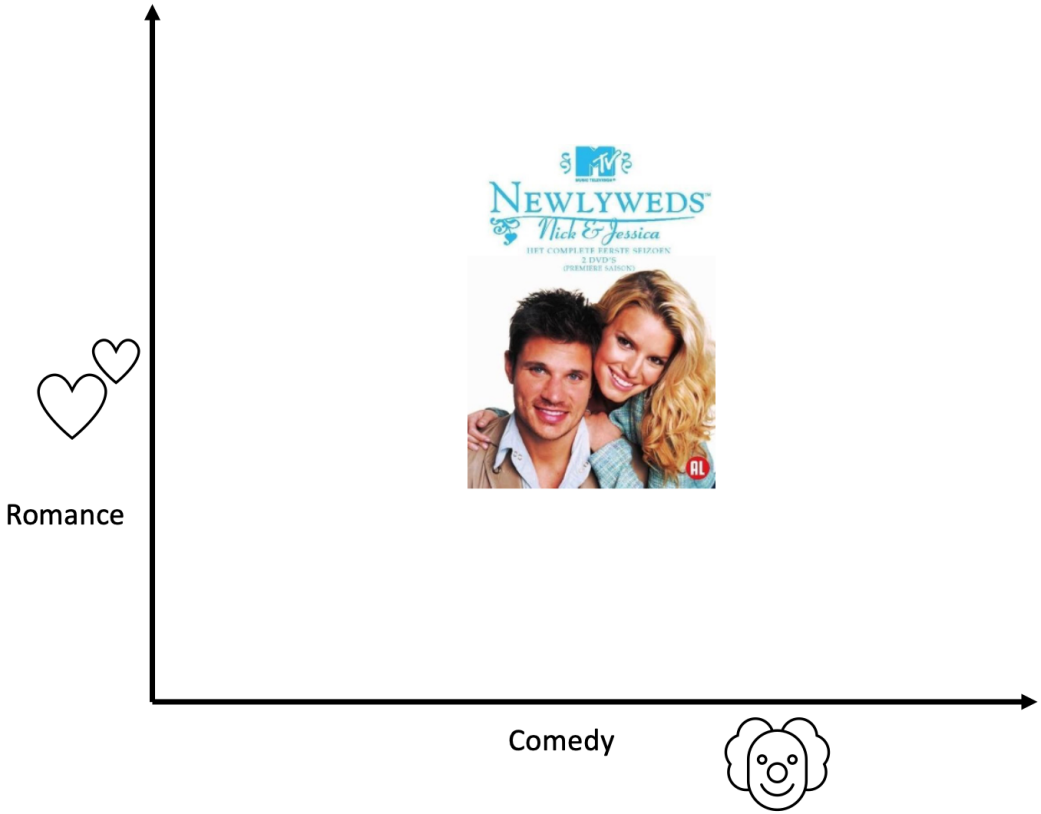
Are there attributes that you could use to estimate similarity in movies?

	genres	title	tagline	release_date	vote_average	vote_count
0	[action, adventure, fantasy, science fiction]	Avatar	Enter the World of Pandora.	2009-12-10	7.2	11800
1	[adventure, fantasy, action]	Pirates of the Caribbean: At World's End	At the end of the world, the adventure begins.	2007-05-19	6.9	4500
2	[action, adventure, crime]	Spectre	A Plan No One Escapes	2015-10-26	6.3	4466
3	[action, crime, drama, thriller]	The Dark Knight Rises	The Legend Ends	2012-07-16	7.6	9106
4	[action, adventure, science fiction]	John Carter	Lost in our world, found in another.	2012-03-07	6.1	2124

Similarity between movies

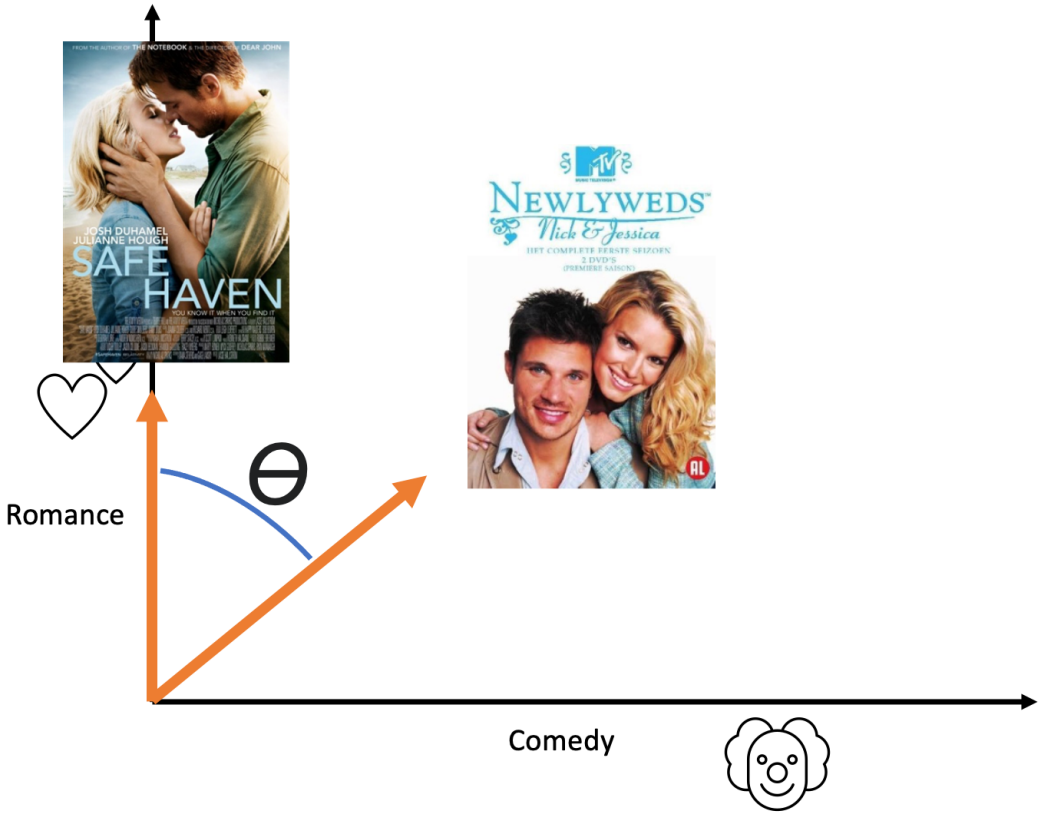


Similarity between movies

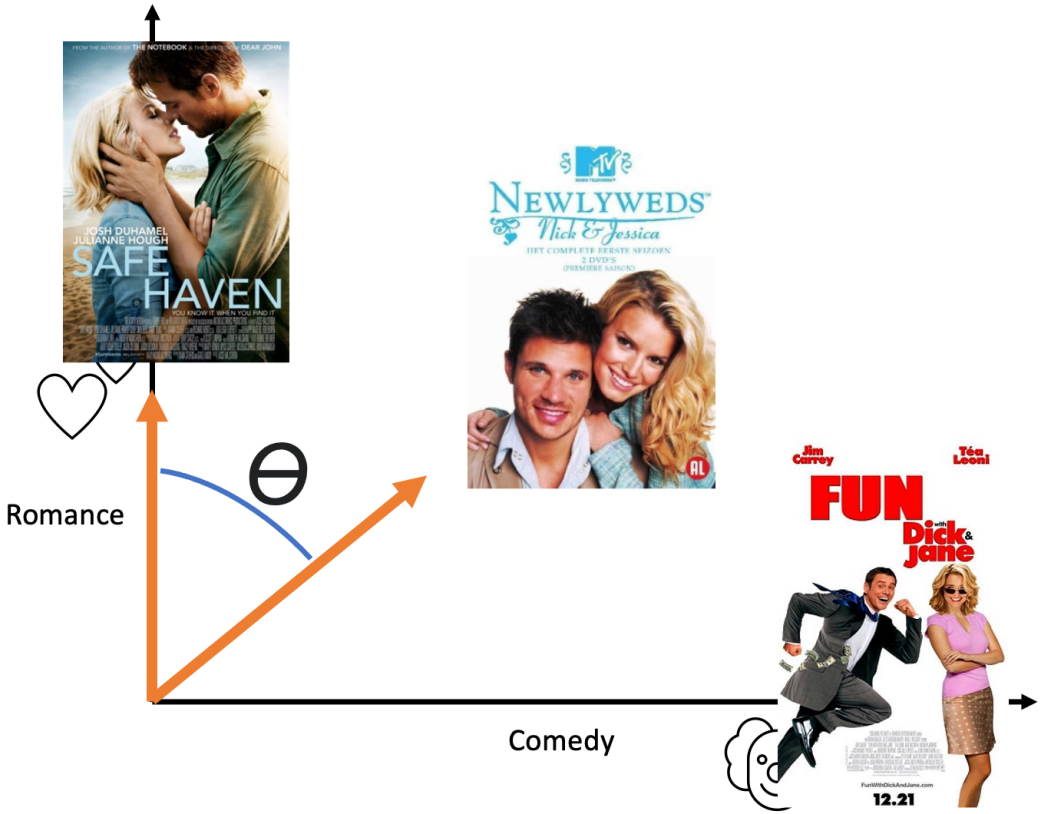


Similarity between movies

Similarity between movies



Similarity between movies



Let's put this in code!

First, create a create a toy dataset.

```
data = data.sample(6)
data[['title', 'genres']]
```

Out[239]:

	title	genres
0	Avatar	action adventure fantasy science fiction
1	Pirates of the Caribbean: At World's End	adventure fantasy action
2	Spectre	action adventure crime
3	The Dark Knight Rises	action crime drama thriller
4	John Carter	action adventure science fiction

Feature selection and vectorization

Let's assume we want to calculate similiarity based on genres. Therefore, we need to vectorize this column.

```
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(data['genres'])
```

Note that we use tfidf vectorizer here, but we you might opt for a different one.

Calculate similiarity

Now, let's calculate cosine similarity scores between the genre attributes of the movies

```
from sklearn.metrics.pairwise import cosine_similarity
sim = cosine_similarity(tfidf_matrix)
```

This returns an array of the similiarity scores between each movie and all other movies.

Cosine Similairity

Let's inspect the output...

```
print(sim)
[[1.         0.34503493 0.         0.         0.40824829 0.        ]
[0.34503493 1.         0.16581288 0.         0.28171984 0.29130219]
[0.         0.16581288 1.         0.25964992 0.         0.56921261]
[0.         0.         0.25964992 1.         0.44115109 0.4561563 ]
[0.40824829 0.28171984 0.         0.44115109 1.         0.        ]
[0.         0.29130219 0.56921261 0.4561563  0.         1.        ]]
```

title	Avatar	Pirates of the Caribbean: At World's End	Spectre	The Dark Knight Rises	John Carter
genre	action adventure fantasy science fiction	adventure fantasy action	action adventure crime	action crime drama thriller	action adventure science fiction
title	genres				
Avatar	action adventure fantasy science fiction	1.000000	0.691870	0.315126	0.084696	0.859850
Pirates of the Caribbean: At World's End	adventure fantasy action	0.691870	1.000000	0.455470	0.122417	0.366490
Spectre	action adventure crime	0.315126	0.455470	1.000000	0.473354	0.366490
The Dark Knight Rises	action crime drama thriller	0.084696	0.122417	0.473354	1.000000	0.098501
John Carter	action adventure science fiction	0.859850	0.366490	0.366490	0.098501	1.000000

Based on this overview, you may assume that users that like *Avatar*, may be interested in *John Carter*. If you want to convert output of cosine_similarity to a df type of object, see here https://github.com/annekroon/CCS-2/blob/main/week06/cosine_to_df.md.

Can we automate this proces?
Can we automatically find the entry with the most similar movie?

Example of a content-based recsys

ge	original_title	...	production_companies	production_countries	release_date	revenue	runtime	spoken_languages	status	tagline	vote_average	vote_count
en	avatar	...	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...	2009-12-10	2787965087	162.0	[{"iso_639_1": "en", "name": "English"}, {"iso...	Released	Enter the World of Pandora.	7.2	11800
en	pirates of the caribbean: at world's end	...	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": "US", "name": "United States o...	2007-05-19	961000000	169.0	[{"iso_639_1": "en", "name": "English"}]	Released	At the end of the world, the adventure begins.	6.9	4500
en	spectre	...	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": "GB", "name": "United Kingdom"...	2015-10-26	880674609	148.0	[{"iso_639_1": "fr", "name": "Fran\u00e7ais"},...	Released	A Plan No One Escapes	6.3	4466
en	the dark knight rises	...	[{"name": "Legendary Pictures", "id": 923}, {"...	[{"iso_3166_1": "US", "name": "United States o...	2012-07-16	1084939099	165.0	[{"iso_639_1": "en", "name": "English"}]	Released	The Legend Ends	7.6	9106
en	john carter	...	[{"name": "Walt Disney Pictures", "id": 2}]	[{"iso_3166_1": "US", "name": "United States o...	2012-03-07	284139100	132.0	[{"iso_639_1": "en", "name": "English"}]	Released	Lost in our world, found in another.	6.1	2124

What are interesting features that you can use to base your recommendation on?

Get the most similar movies

First, create an array of cosine scores

```
data['combined_features'] = data[['original_title', 'genres', 'overview', 'tagline']]
.apply(lambda x: ','.join(x.dropna().astype(str)),axis=1)
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(data['combined_features'])
cosine_sim = cosine_similarity(tfidf_matrix)
```

Let's say we want to find the similarity scores associated with the movie `The Dark Night Rises`. Just by looking at the dataframe, we know it is at index 3.

Example of a content-based recsys

```
data.head()
```

	movie_id	title	cast	crew	budget	genres	homepage	keywords	original_language	origin
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	[{"id": 1463, "name": "culture clash"}, {"id":...	en	
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	pirate caribl wor
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	1 knig

```
cosine_sim[3]
```

```
array([0.02331349, 0.00399197, 0.00780174, ..., 0.02860083, 0.03756737, 0.018984  ])
```

A more systematic way to get the index value, is to simple look it up:

```
indices = pd.Series(data.index, index = data['original_title'])
index = indices['the dark knight rises']
print(index)
```

```
3
```

Get the most similar movies

- Next, we need to sort the associated vector of cosine similarity scores for this movie to get the most similair movies.
- Before sorting the cosine scores, we map the movie-index to the cosine value. We can do so by simple enumerating the cosine scores:

```
sim_scores = list(enumerate(cosine_sim[index]))
```

```
[(0, 0.023313489832942038),
(1, 0.003991972883233364),
(2, 0.007801739082093903), ...
```

Next, we can simple sort the cosine scores:

```
sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
sim_scores = sim_scores[1:11]
```

```
[(3, 0.9999999999999998),
(65, 0.3484608502957436),
(299, 0.32460952432731993),
```

Get the most similar movies

And finally look up the associated movies..

```
movie_indices = [i[0] for i in sim_scores]
movie_indices
```

```
[299, 1359, 3854, 428, 119, 210, 9, 2507, 979]
```

Map the index values to the dataframe:

```
data.iloc[movie_indices]['title']
```

299	Batman Forever
1359	Batman
3854	Batman: The Dark Knight Returns, Part 2
428	Batman Returns
119	Batman Begins
210	Batman & Robin
9	Batman v Superman: Dawn of Justice
2507	Slow Burn
979	Free State of Jones

Feature selection and engineering

- Feature engineering is very important here. What qualities or attributes do we want to incorporate?
- Think about this critically. Instead of using genres or authors, you could think about crafting features yourself, such as topics.

how can we identify similar items?

1. Cosine similarity
2. Soft cosine similarity
3. LDA: topic scores

Benefits

- Content-based recommender systems can be very efficient...
- They are often part of more complex recommender systems that leverage (deep) supervised learning

Drawbacks

- Features that are not part of the user profile will be neglected; e.g., if the user does not like Super Hero movies, the recommender system wil never recommend this.
- does not use the power community data. Recommendations may be obvious (e.g., *Harry Potter* recommendation when you like *Lord of the Rings*)

Collaborative RecSys

Collaborative filtering

What are collaborative systems?

- Tries to overcome some of the limitations of content-based systems
- Leverages the power of the community, tries to give relevant, but also suprising recommendations.
- Very succesful models in industry settings

Types of collaborative systems

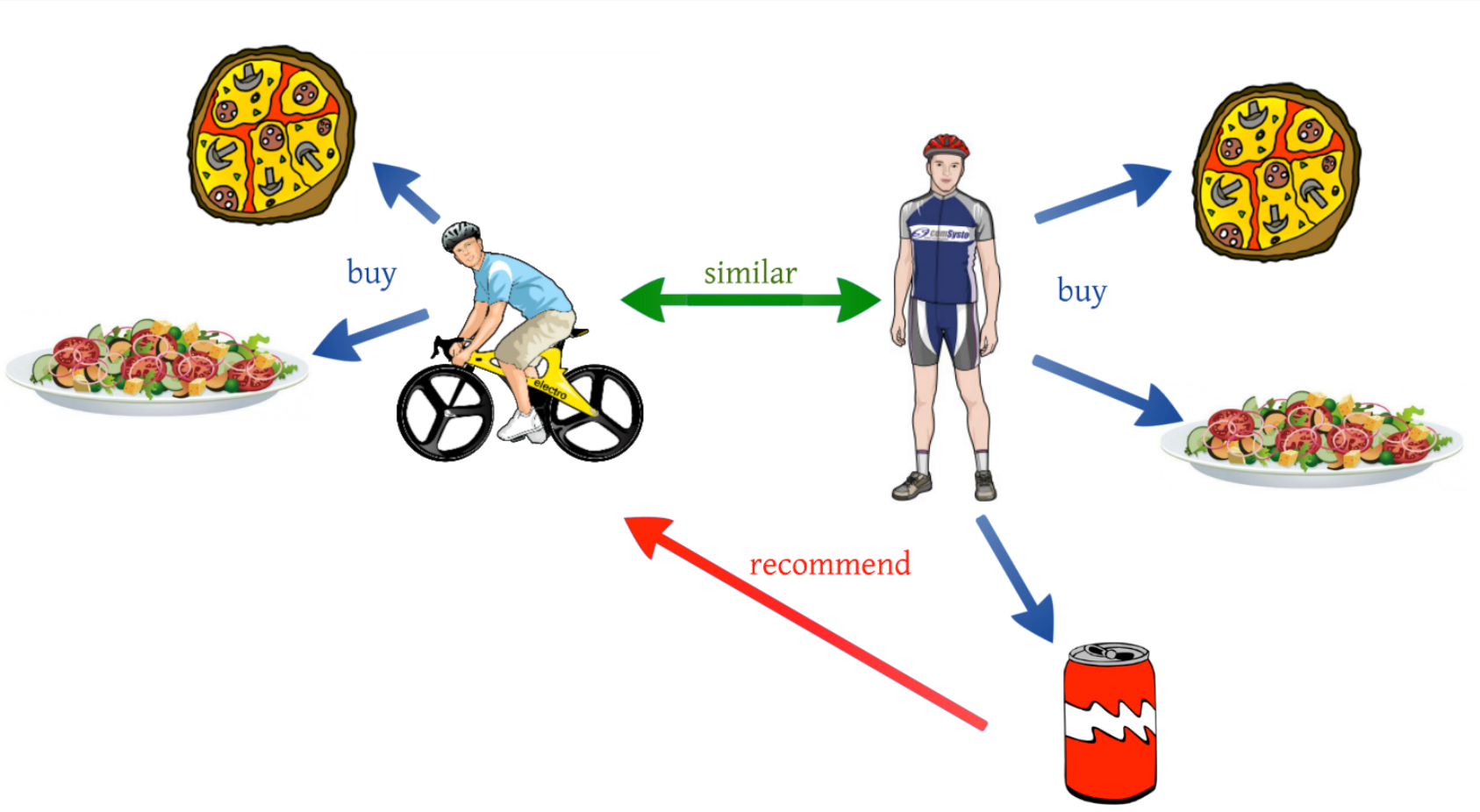
1. User-based collaborative filtering
2. Item-based collaborative filtering

User-based filtering

Similar users...

- If we find users that are similar in terms of the things they liked in the past...
- ...we can use this information to predict what they like in the future

User-based filtering



[1] Source: https://medium.com/@akhilesh3091999/recommender-system-bb032b16ab67

Item-based filtering

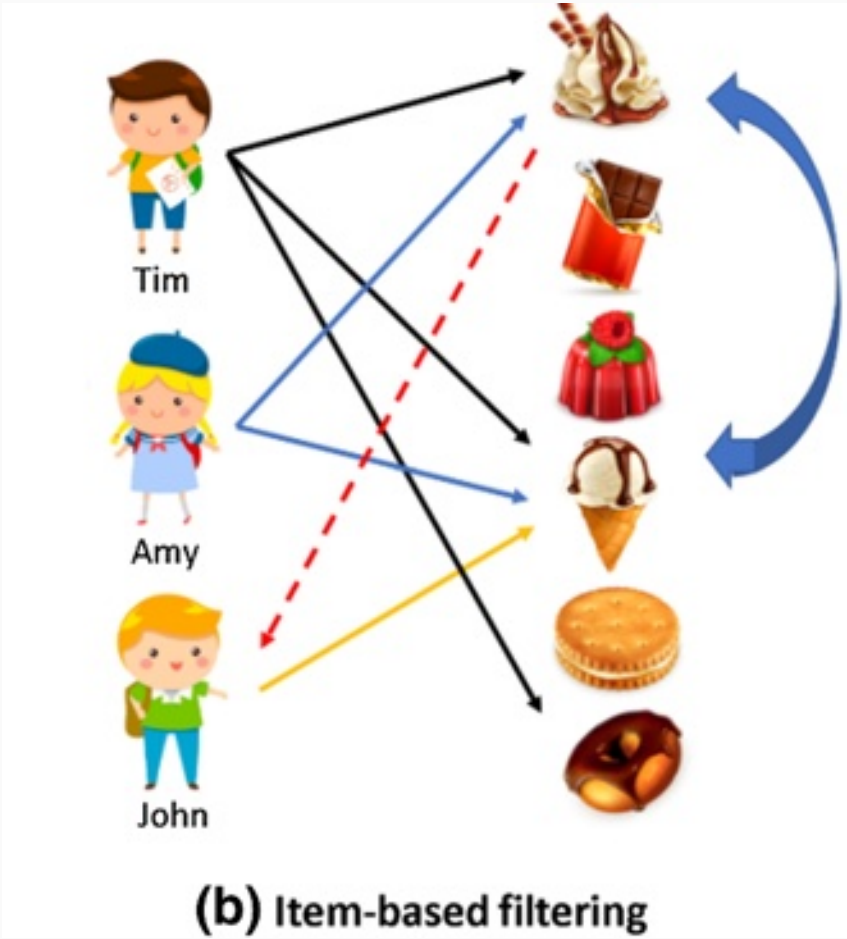
Similar items...

- If two items is rated similarly by a group of people
- ...these product might be similar

Example...

- If Alex, Sanne and Marthe like skincare product X and Y, and Denise buys skincare product Y, X will be recommended.

User-based filtering



[2] Source: https://predictivehacks.com/item-based-collaborative-filtering-in-python/

Example: Amazon

Group ass

Build your own recommender system

Part of the assignment: Build a recommender (20% of final grade)

- Build a recommender system, based on the insights from this week (week 6). It's up to you to decide whether you build a knowledge-based or content-based recommender system.
- Think about relevant features that you want to use in your algorithm design. Based on which features do you want to recommend content?

Build your own recommender system

Practice with the materials!

- To be able to do this correctly, it is essential that you understand the code of this week's lab session.
- Carefully walk through this week's assignment, and to whether questions arise.
- It's up to you to decide whether you want to build a simple knowledge-based or content-based recommender system. Base your selection on the available data columns.

References

References


Kim, H. S., Yang, S., Kim, M., Hemenway, B., Ungar, L., & Cappella, J. N. (2019). An experimental study of recommendation algorithms for tailored health communication. *Computational Communication Research*, *1*(1), 103–129. https://doi.org/10.5117/ccr2019.1.005.sukk


Locherbach, F., & Trilling, D. (2018). 3bij3: A framework for testing effects of recommender systems on news exposure. *Proceedings - IEEE 14th International Conference on eScience, e-Science 2018*, 350–351. https://doi.org/10.1109/eScience.2018.00093


Möller, J., Trilling, D., Helberger, N., & van Es, B. (2018). Do not blame it on the algorithm: an empirical assessment of multiple recommender systems and their impact on content diversity. *Information Communication and Society*, *21*(7), 959–977. https://doi.org/10.1080/1369118X.2018.1444076


Wieland, M., Von Nordheim, G., & Kleinen-Von Königslöw, K. (2021). One recommender fits all? An exploration of user satisfaction with text-based news recommender systems. *Media and Communication*, *9*(4), 208–221. https://doi.org/10.17645/mac.v9i4.4241