Project 4:

Movie recommendation

By Group 5

Introduction - Topic

- Our aim was to solve a problem facing many people around the world today what to watch?

- With streaming services offering hundreds of thousands of movies we spend more time figuring out what to watch then actually enjoying a movie

Introduction - Our Aim

- To utilize machine learning to give movie recommendations so we never need to debate what movie looks the most interesting.

- Our approach utilizes two machine learning models, a K-nearest neighbor (KNN) model and a Neural Network (NN).

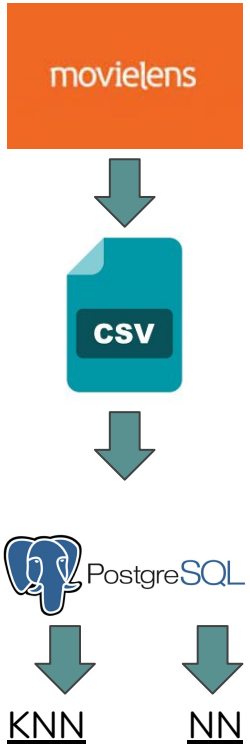
- The KNN provides a simple yet accurate model.
- The NN provides a detailed model.

Introduction - Our Approach

- We decided to use a smaller version of the movielens data set, found here: https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset

- Then we had used PostgreSQL for our database

- From there we could build the KNN model and the NN model



About the data

There were two datasets we had used for this project:

Movies.csv

movieID	title	genres
An id for each movie	Titles and year for each movie	A list of the movies genres

Users.csv

userID	movieID	rating	timestamp
An id for each user	An id for each movie	Each reviews rating	A timestamp of when the user left the review

In total we had:

- 9724 Movies
- 610 Users
- 100,836 Reviews

Prepare Your Database

- ➢ Create tables for datasets in the movies_db database
  - movies
  - ratings
- ➢ Primary Key
  - movie_id
- ➢ Select and inner join the tables

Note: The tables created doesn't have data at this stage

Extract

- Read the data using pandas library and display the dataframes.

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091

Transform

- Create a copy of the filtered dataframes from specific columns
- Rename the column headers
- Display the new dataframes

	user_id	movie_id	rating	timestamps
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402

Load

- Connect with the local database
- Inspect tables
- Load the data into the database by using .to_sql method
- Query both tables to confirm if the data has been added

	user_id	movie_id	rating	timestamps
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815

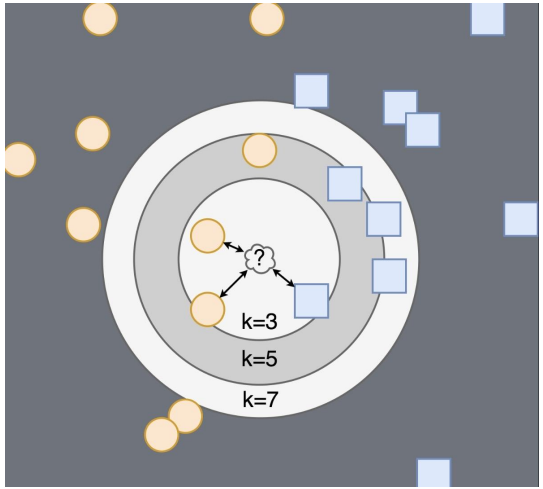
Join and Preview

- Time to inner join in pandas
- Preview the sql database

	movie_id bigint	title text	genres text	user_id bigint	rating double precision	timestamps bigint
1	1	Toy Story ...	Adventur...	1	4	964982703
2	3	Grumpier ...	Comedy|...	1	4	964981247
3	6	Heat (199...	Action|Cri...	1	4	964982224
4	47	Seven (a....	Mystery|T...	1	5	964983815

KNN learning Model

We decided to use the KNN model because of it’s ability to achieve accuracy in a wide variety of prediction problems. We used sklearn,Matplotlib and numpy.
Our data set had over 100 000 movie reviews for over 9 000 movies.



We created a KNN model that could help recommend movies based on ratings. Before creating the model we had to clean the data to make sure that we didn't have any duplicates and also exclude movies with very high ratings compared to others in the dataset to reduce bias.

	movie_id	title	genres	user_id	rating	timestamps
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	214	1	853937855
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	213	1	1316196157
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	153	0	1525548642
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	89	1	1520408314
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	233	1	1524781249
...	...	...	...	...	...	...
100831	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy	184	1	1537109082
100832	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy	184	1	1537109545
100833	193585	Flint (2017)	Drama	184	1	1537109805
100834	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation	184	1	1537110021
100835	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy	331	1	1537157606

100836 rows × 6 columns

+ Code + Markdown

```
[14]
# Look at movie value counts for cleanup
review_count = data_df['movie_id'].value_counts()
review_count
```

```
356       329
318       317
296       307
593       279
2571      278
         ...
4093        1
4089        1
58351       1
4083        1
193609      1
Name: movie_id, Length: 9724, dtype: int64
```

```
[15]
data_df = data_df[~data_df['movie_id'].isin(movies_to_exclude)]
# Check to make sure cleaning was successful
data_df['movie_id'].value_counts()
```

```
356       329
318       317
296       307
593       279
2571      278
         ...
5329       10
5296       10
68793      10
102481     10
45668      10
Name: movie_id, Length: 2269, dtype: int64
```

```
[16]
# Check remaining ratings
data_df
```

	movie_id	title	genres	user_id	rating	timestamps
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	214	3.0	853937855
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	213	3.5	1316196157
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	153	2.0	1525548642
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	89	3.0	1520408314
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	233	3.0	1524781249
...	...	...	...	...	...	...
100792	187593	Deadpool 2 (2018)	Action|Comedy|Sci-Fi	249	5.0	1531611534
100793	187593	Deadpool 2 (2018)	Action|Comedy|Sci-Fi	248	4.5	1534599909
100794	187593	Deadpool 2 (2018)	Action|Comedy|Sci-Fi	25	5.0	1535470534
100795	187593	Deadpool 2 (2018)	Action|Comedy|Sci-Fi	98	5.0	1532457913
100796	187593	Deadpool 2 (2018)	Action|Comedy|Sci-Fi	233	2.5	1536968163

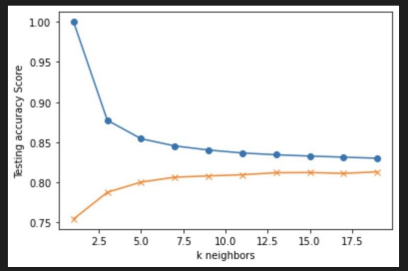
81116 rows × 6 columns

We converted the rating column to be either a recommended or non recommended so that we could use it as the target for testing the model. In order to arrive at a model with an acceptable performance, we took some optimisation steps based on different rating cut off values 3,5 and 2.5.

```
plt.xlabel("k neighbors")
plt.ylabel("Testing accuracy Score")
plt.show()
```

[20]

```
k: 1, Train/Test Score: 1.000/0.754
k: 3, Train/Test Score: 0.877/0.788
k: 5, Train/Test Score: 0.854/0.800
k: 7, Train/Test Score: 0.845/0.806
k: 9, Train/Test Score: 0.840/0.808
k: 11, Train/Test Score: 0.837/0.810
k: 13, Train/Test Score: 0.834/0.812
k: 15, Train/Test Score: 0.833/0.812
k: 17, Train/Test Score: 0.831/0.811
k: 19, Train/Test Score: 0.830/0.813
```

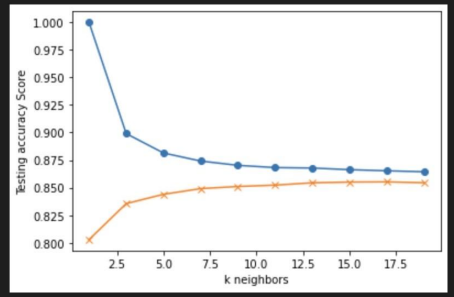



```
# Note that k: 3 provides the best accuracy where the classifier starts to stablize
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
print('k=3 Test Acc: %.3f' % knn.score(X_test_scaled, y_test))
```

[21]

```
k=3 Test Acc: 0.788
```

```
k: 1, Train/Test Score: 1.000/0.803
k: 3, Train/Test Score: 0.899/0.835
k: 5, Train/Test Score: 0.881/0.844
k: 7, Train/Test Score: 0.874/0.849
k: 9, Train/Test Score: 0.870/0.851
k: 11, Train/Test Score: 0.868/0.852
k: 13, Train/Test Score: 0.868/0.854
k: 15, Train/Test Score: 0.866/0.855
k: 17, Train/Test Score: 0.865/0.855
k: 19, Train/Test Score: 0.864/0.854
```



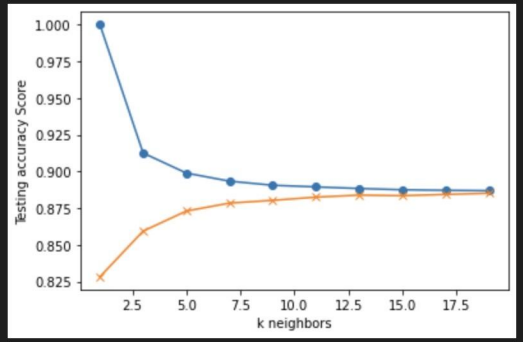

```
# Note that k: 3 provides the best accuracy where the classifier starts to stablize
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
print('k=3 Test Acc: %.3f' % knn.score(X_test_scaled, y_test))
```

[27]

```
k=3 Test Acc: 0.835
```

This paid of as we ended up with a model that demonstrates meaningful predictive power of 85.9%

```
k: 1, Train/Test Score: 1.000/0.828
k: 3, Train/Test Score: 0.913/0.859
k: 5, Train/Test Score: 0.899/0.873
k: 7, Train/Test Score: 0.893/0.878
k: 9, Train/Test Score: 0.891/0.880
k: 11, Train/Test Score: 0.889/0.882
k: 13, Train/Test Score: 0.888/0.884
k: 15, Train/Test Score: 0.888/0.883
k: 17, Train/Test Score: 0.887/0.884
k: 19, Train/Test Score: 0.887/0.885
```



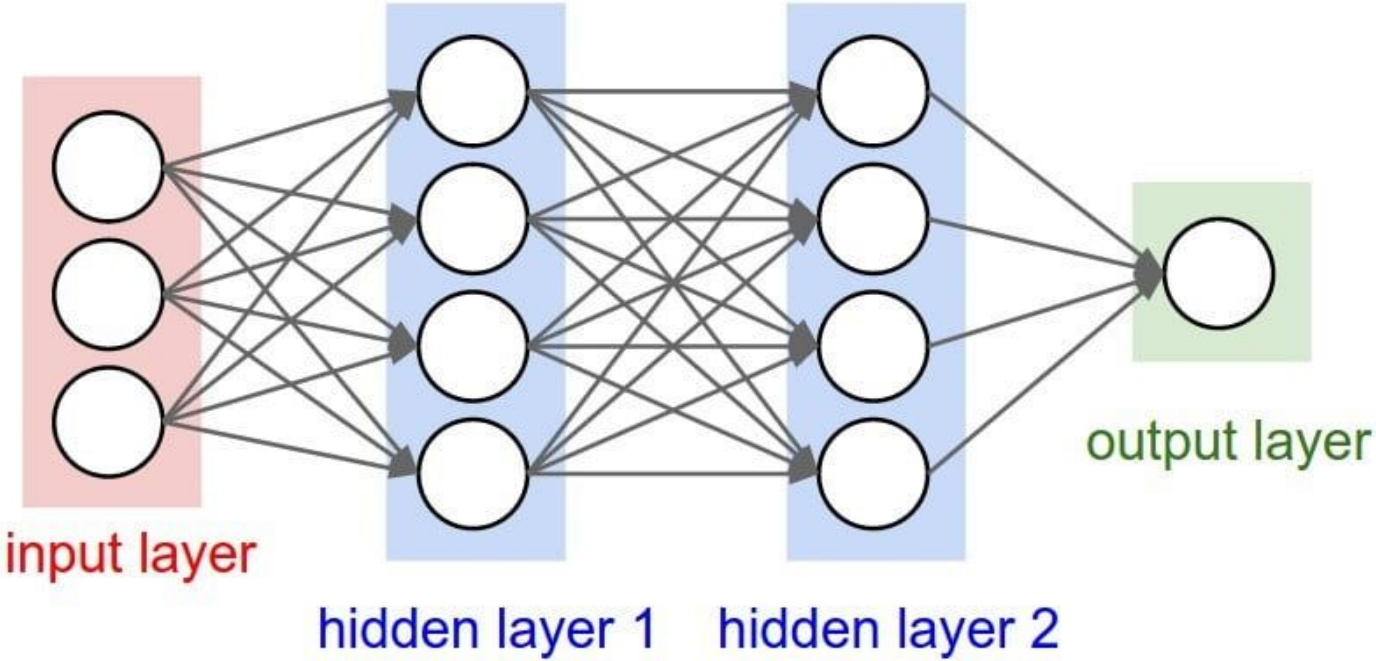

```
# Note that k: 3 provides the best accuracy where the classifier starts to stablize
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
print('k=3 Test Acc: %.3f' % knn.score(X_test_scaled, y_test))
```

[33]

```
k=3 Test Acc: 0.859
```

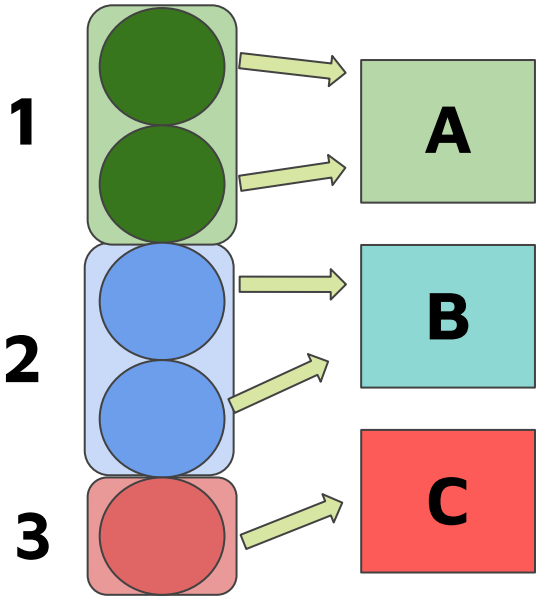
Deep Learning

Neural Network

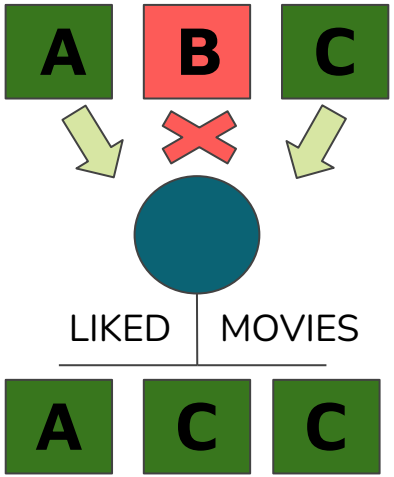


What to use as inputs? / What type of model?

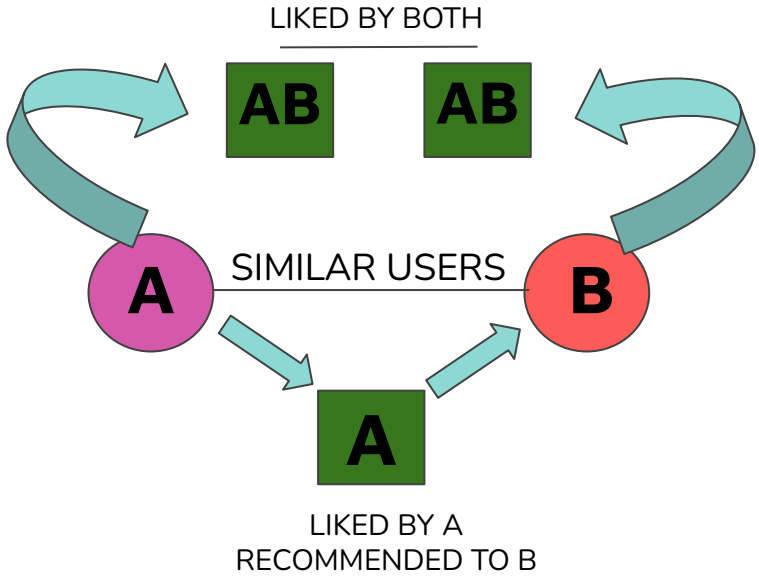
Demographic Filtering



Content-based Filtering

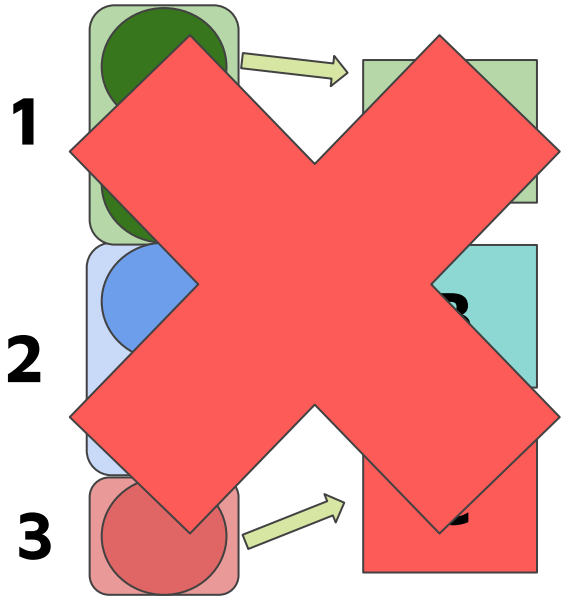


Collaboration-based Filtering

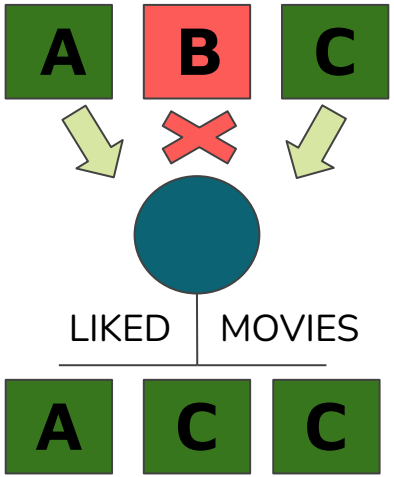


What to use as inputs? / What type of model?

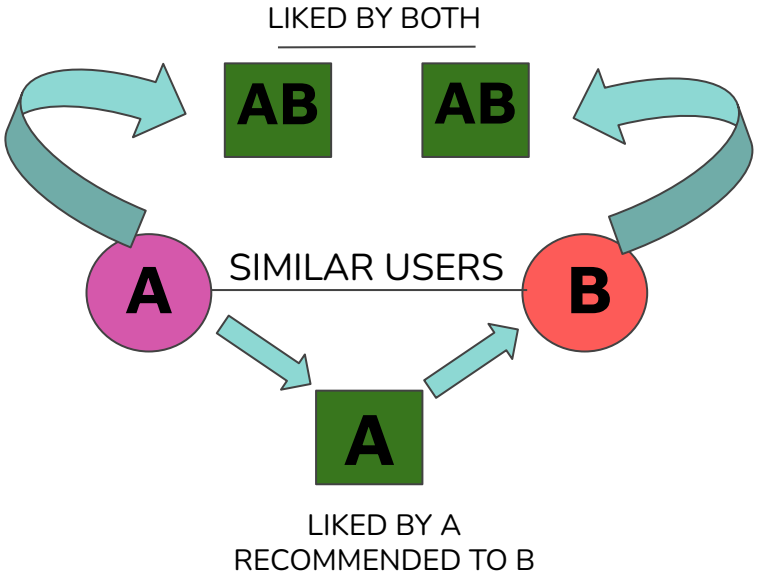
Demographic Filtering



Content-based Filtering



Collaboration-based Filtering

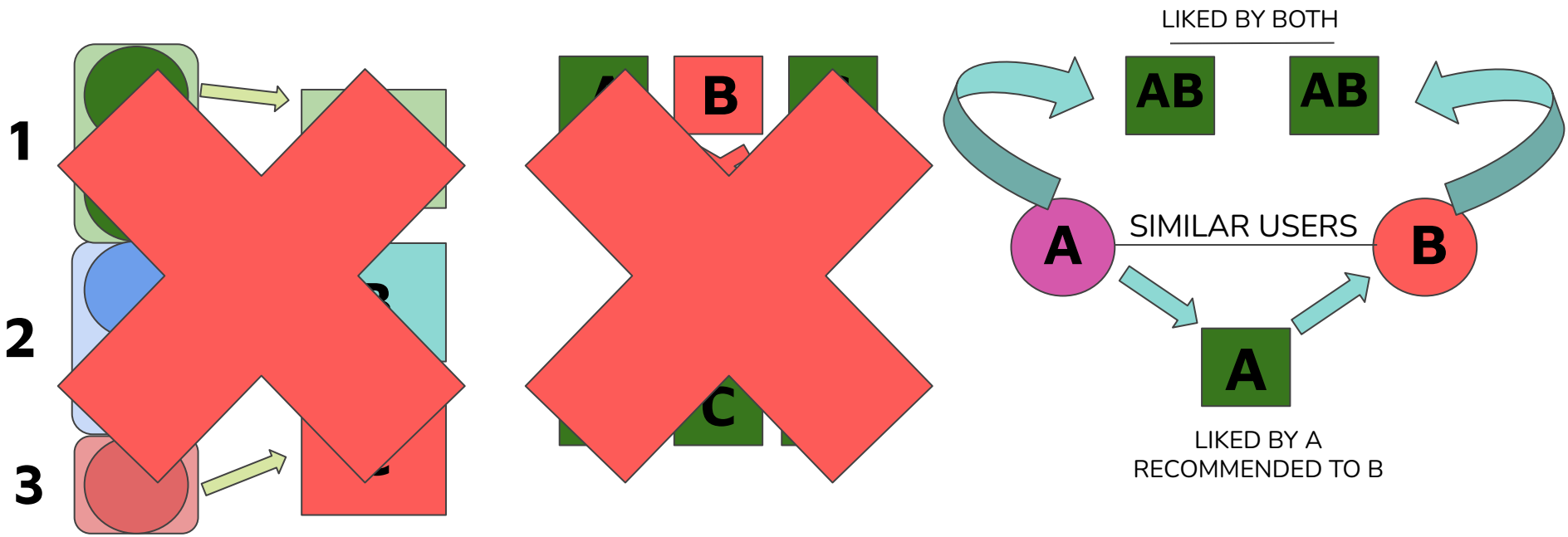


Neural Collaborative Filtering

Demographic Filtering

Content-based Filtering

Collaboration-based Filtering

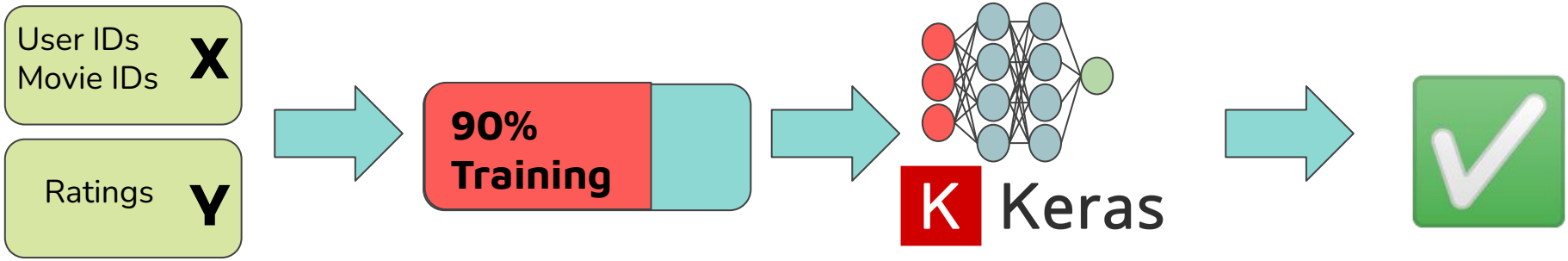


Neural Collaborative Filtering - Libraries Used

Keras

Neural Collaborative Filtering is a paper from NExT a nationally funded research foundation in Singapore that can be found here: https://arxiv.org/abs/1708.05031

Neural Collaborative Filtering - What was the process



MAKING VARIABLES

SPLIT THE DATA

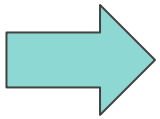
BUILD AND FIT MODEL

READY TO PREDICT

Neural Collaborative Filtering - Predictions

USER ID 315

title	genres
Beauty of the Day (Belle de jour) (1967)	Drama
Wild Bunch, The (1969)	Adventure|Western
Breakfast at Tiffany's (1961)	Drama|Romance
Apartment, The (1960)	Comedy|Drama|Romance
My Fair Lady (1964)	Comedy|Drama|Musical|Romance
2001: A Space Odyssey (1968)	Adventure|Drama|Sci-Fi
Mary Poppins (1964)	Children|Comedy|Fantasy|Musical
Sound of Music, The (1965)	Musical|Romance
Bonnie and Clyde (1967)	Crime|Drama
Good, the Bad and the Ugly, The (Buono, il bru...	Action|Adventure|Western



MOVIES FOR USER 315

title	genres
Princess Bride, The (1987)	Action|Adventure|Comedy|Fantasy|Romance
Austin Powers: International Man of Mystery (1...	Action|Adventure|Comedy
Starship Troopers (1997)	Action|Sci-Fi
NeverEnding Story, The (1984)	Adventure|Children|Fantasy
Serpico (1973)	Crime|Drama
Ice Age (2002)	Adventure|Animation|Children|Comedy
Die Another Day (2002)	Action|Adventure|Thriller
Body of Evidence (1993)	Drama|Thriller
Holiday, The (2006)	Comedy|Romance
Proposal, The (2009)	Comedy|Romance

Neural Collaborative Filtering - Flaws

DATA

- Old Dataset
  - September 2018
- Small Dataset
  - Full Dataset : 62,000 movies, 162,000 users, 25 million ratings
  - Our Dataset: 9742 movies, 610 users, 100836 ratings

RESULTS

- High Data Loss
  - 62.99%
- Low Accuracy
  - 13.57%
- Moderate F1 Score
  - 66.79%

Neural Collaborative Filtering - Retrospect

Collaborative Filtering is hard

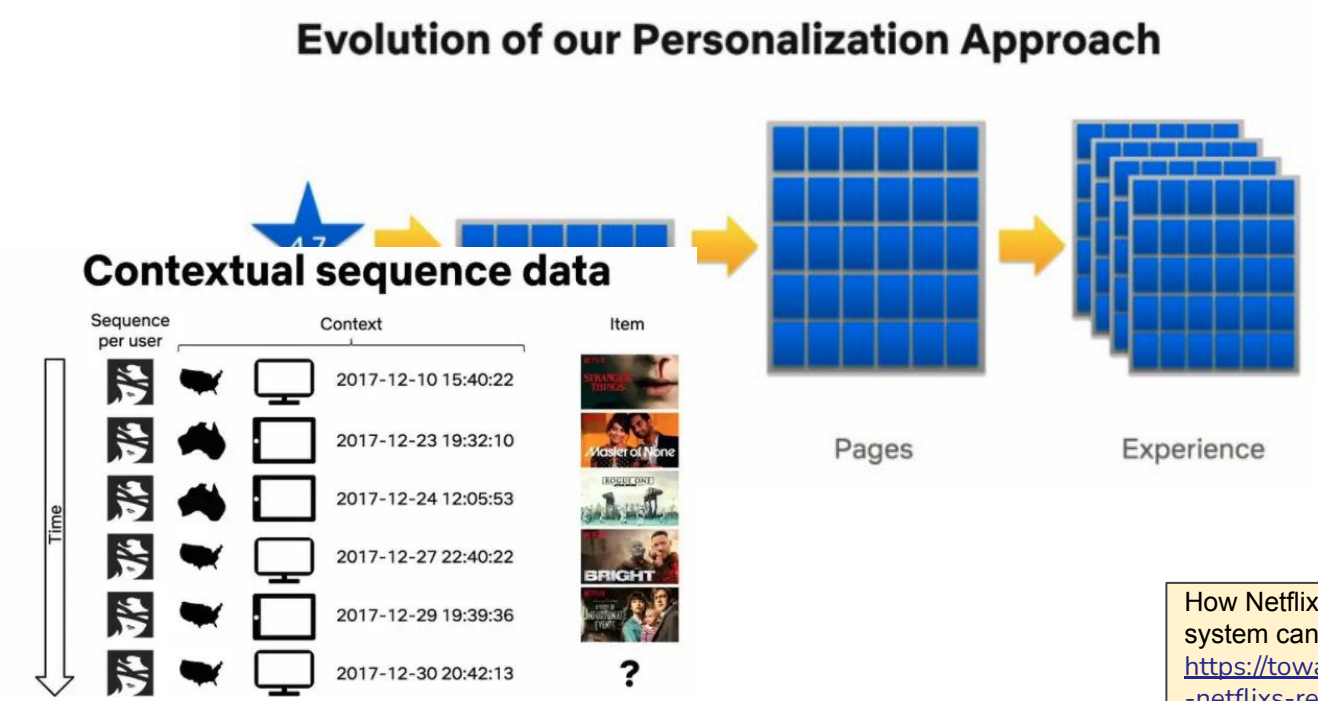
- Needs lots of data
- Needs lots of processing power

How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

Neural Collaborative Filtering - Retrospect

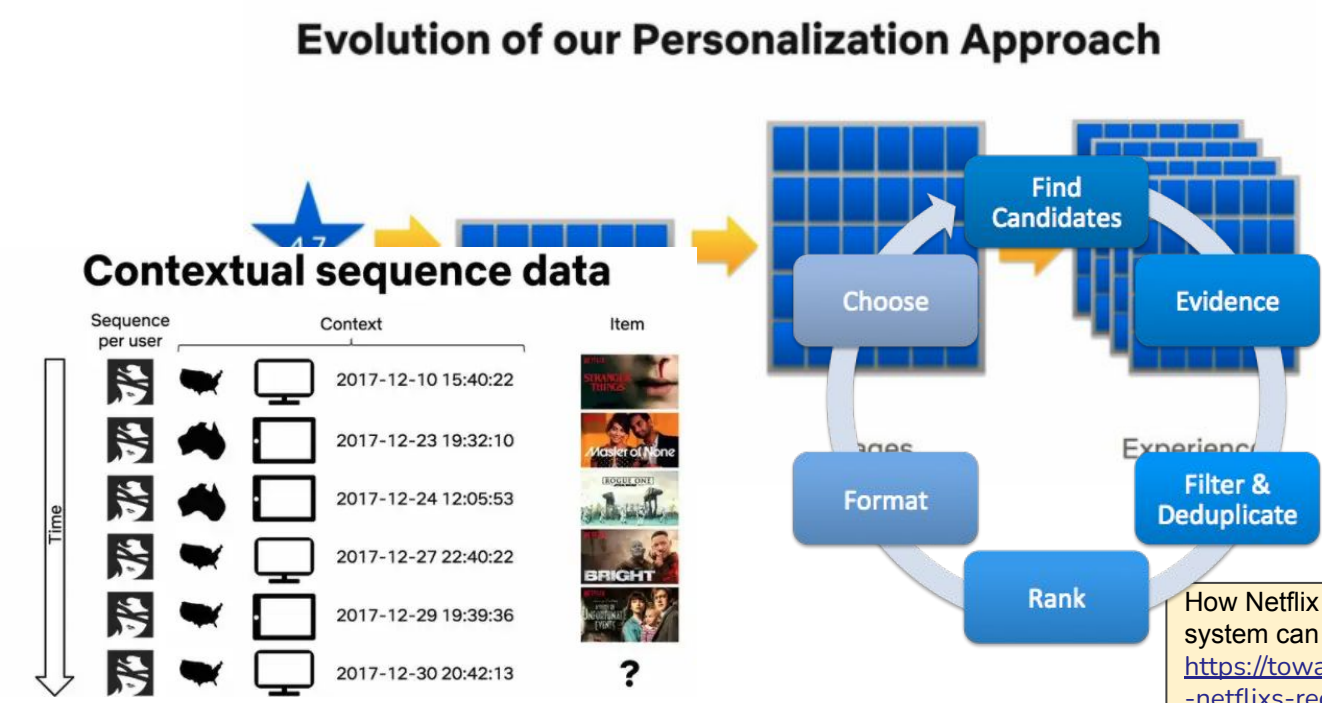
How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

Neural Collaborative Filtering - Retrospect



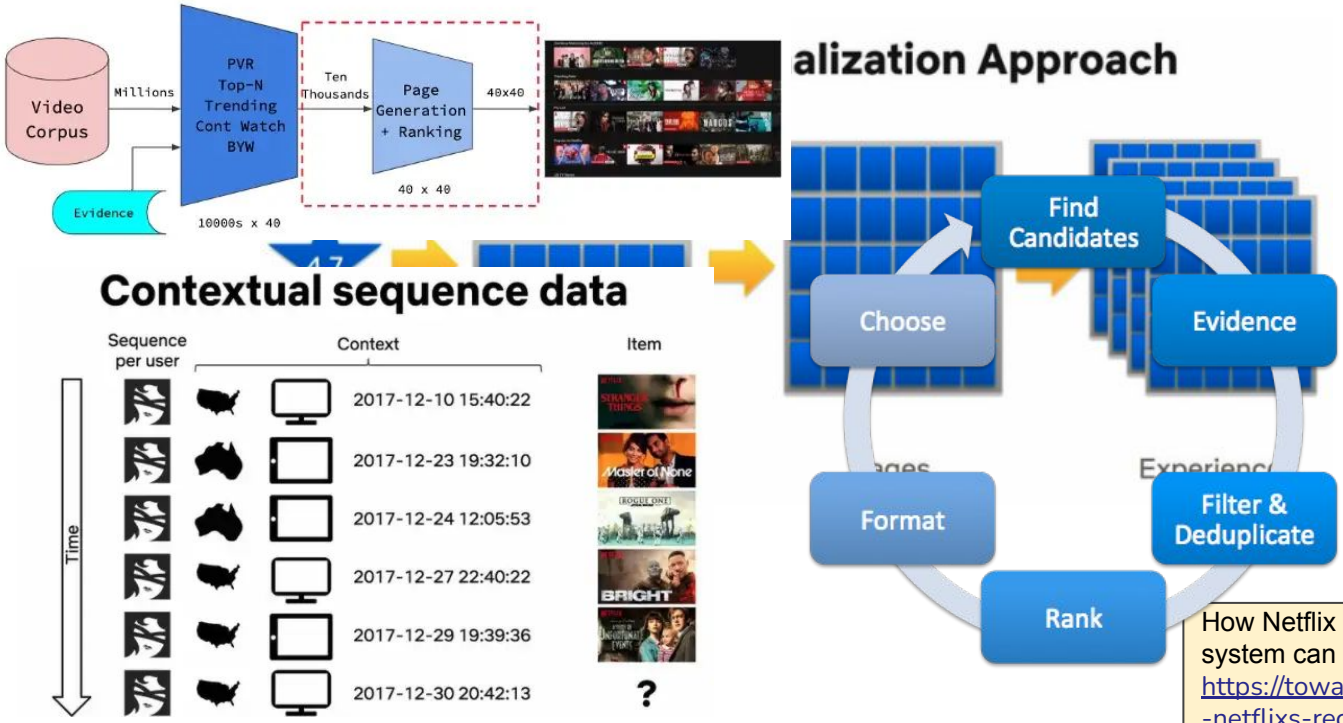
How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

Neural Collaborative Filtering - Retrospect



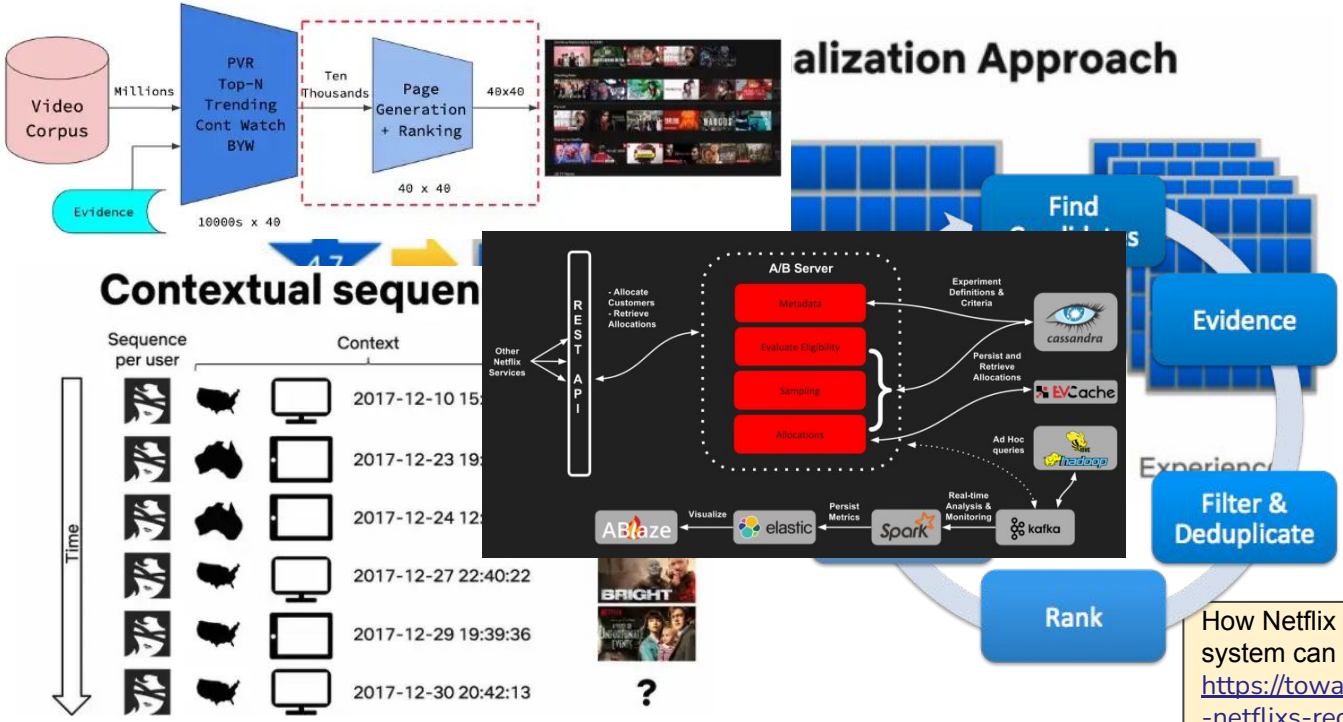
How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

Neural Collaborative Filtering - Retrospect



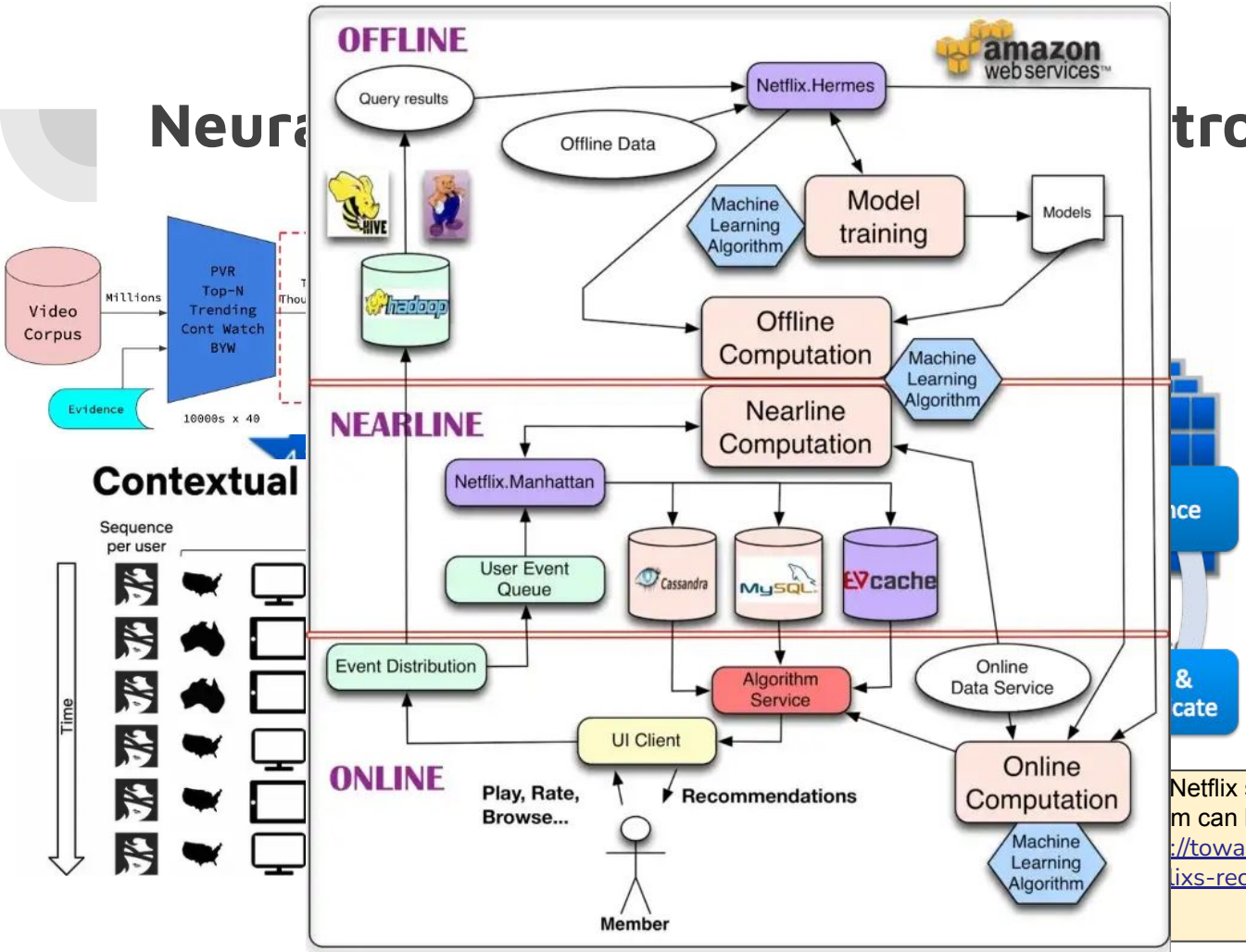
How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

Neural Collaborative Filtering - Retrospect



How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

Neura trospect



Netflix structures there recommendation
m can be found here:
://towardsdatascience.com/deep-dive-into
ixs-recommender-system-341806ae3b4

Neural Collaborative Filtering - Retrospect

Collaborative Filtering is hard

- Needs lots of data
- Needs lots of processing power

How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

Neural Collaborative Filtering - Retrospect

Collaborative Filtering is hard but **fun**

- We had still managed to make a small model
- It still somewhat functioned and provided good data
- Next time I will aim to pick something within my scope!

How Netflix structures there recommendation system can be found here: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48

What we had achieved

- We had successfully built a database in PostgreSQL using an ETL method
- Using the data from the database we built two machine learning models:
  - An accurate KNN model
  - A NN model that could give recommendations

Thank you for listen and Congratulations on completing the boot camp!