DAT410: Assignment 2

Group 60: Cecilia Nyberg, Elin Stiebe

January 2024

1 Task 1: Reading and reflection

Read the papers "The Netflix Prize Download The Netflix Prize" and "Lessons from the Netflix Prize Challenge" and write a half-page summary of your takeaways from them. Pick two design features (e.g., regarding data, models, algorithms) that characterize the task and the solution in the papers and discuss how they may differ in another application of recommendation systems.

Both Bell and Koren, 2007 and Bennett and Lanning, 2007 describe the competition Netflix started within the ML community which was challenged to improve its recommendation system. The first mentioned article more concretely writes that they urged the community to improve their recommendation algorithm by reducing their RMSE by 10%. After creating such an algorithm the programmer or company was supposed to post the code online. The winner would get a big cash prize. As the prize suggests, the articles further describe just how important Netflix's recommendation algorithm is for their business. Users who get recommended movies in line with their tastes tend to stay on the site. As mentioned in Bennett and Lanning, 2007, "If subscribers fail to find movies that interest and engage them, they tend to abandon the service.".

A specific feature of the Netflix recommendation system is the chosen data. The data consists of 100 million reviews. The prize dataset was created by randomly choosing subscribers with no less than 20 ratings. One challenge with the data is the large number of users who have never rated any movie. The article also mentions the importance of the models' data, the models don't only need information about each movie's ratings but also what movie each user rated. Another aspect and feature of the data that the Netflix recommendation algorithm takes into account is that it is best to be kept as a whole. Deleting a certain row would result in deleting the possibility to recommend a user a movie and thus to a direct financial loss. This makes the problem even more complex. In other problems, some data points might be deleted to make the problem easier to solve. Every situation is unique in its own way!

Another specific feature is regarding the model. According to Bell and Koren, 2007, all models which rely on complete data had to be abandoned when they developed the recommendation system. The winner of the challenge was AT&T. According to Bell and Koren, 2007, they relied on ensembles with models that were supposed to complete each other's weaknesses. The takeaway of the winner's strategy is to always consider what data to take into account, allow for models that fit the data, and use them in ensembles when needed to minimize the risk of errors. For a smaller-scaled recommendation system, using an ensemble could instead make the system worse since it introduces unnecessary complexity, making the system harder to manage and more time-consuming to run.

2 Task 2: Implementation

After reading the Netflix articles one could want to consider the following for recommending a movie to a user:

- Ratings for a specific movie.
- The genre preferences of the user compared to the movie.
- How similar users have rated other movies.

The algorithm proposed can be seen in the appendix and only utilizes points one and two in the list above. The idea is to recommend movies based on what movies each person has rated and what genres that movie has. The algorithm takes one user at a time and calculates the top three genres for that person. This is done by summarizing the ratings for each genre and choosing the three genres with the highest value. Then five movies are selected that have all these three genres and that the user has not watched already. Below are the recommendations for Vincent, Edgar, Addilyn, Marlee, and Javier.

- For Vincent, we recommend: Supercross, Crouching Tiger, Hidden Dragon, Sands of Iwo Jima, El Mariachi, Kites, since Vincent likes: action, drama, romance
- Edgar, we recommend: Ghost, El Mariachi, Punch-Drunk Love, Match Point, Out of Sight, since Edgar likes: thriller, romance, drama
- For Addilyn, we recommend: Novocaine, Punch-Drunk Love, Made, Wasabi, Chill Factor, since Addilyn likes: thriller, comedy, drama
- For Marlee, we recommend: Novocaine, Punch-Drunk Love, Made, Wasabi, Chill Factor, since Marlee likes: comedy, thriller, drama
- For Javier, we recommend: Beverly Hills Chihuahua, The Magic Sword: Quest for Camelot, Growing Up Smith, Hannah Montana: The Movie, Mrs. Doubtfire, since Javier likes: family,

drama, comedy

There are some strengths and weaknesses with the chosen algorithm. One strength is that a user probably likes movies with their top three genres since they have a high sum of rankings for those genes. It either means that the user watches a lot of movies within these genres or ranks movies within these genres highly. One weakness is that all other preferences besides genres are disregarded. For example, if a user prefers a certain actor. Another weakness is that the algorithm only accounts for the ratings, not the watched movies. Ratings might reflect what movies the user wants to like but not actually what the user usually chooses to watch.

Further, the algorithm considered two lower ratings within the same genre as equivalent to one higher rating within another genre. For example, one 5-star rating within one genre is equivalent to five 1-star ratings within another genre. This could be both a pro or con depending on the goal of the company and algorithm. Is the goal to have customers viewing many movies of a few highly rated ones? One last con is that comparing users with other similar users is not utilized. This could be valuable insight and could be used for example by choosing a high-rated movie from a similar user that the current user hasn't seen but might like.

3 Task 3: Discussion

Assessing the quality of a recommendation system before deploying it to users is difficult. Why? In a few paragraphs, discuss fundamental challenges in the evaluation of recommendation systems and how they may lead to problems in practice.

It is hard to assess the quality of a recommendation system before deployment since the quality is most often measured in terms of the error between the prediction and the actual score(in the case of Netflix, the predicted rating and the actual rating). This can be done with a test set of data before deployment, though these types of recommendation systems are inherently time-sensitive. To improve on the recommendation system one needs to rely on input from the users.

Another challenge is the feedback loop. Sometimes it takes a long time to see the effects of the implemented algorithm. This also depends on the objective of the recommendation algorithm. It might also be a challenge to distinguish the exact contribution of the algorithm in relation to other business decisions made during the same time.

An additional challenge is that if the recommendation system is based on ratings, it might not reflect what the user chooses to watch and the effect might therefore be different than expected. For example, someone might rate documentaries highly because they want to be a person who likes documentaries, but in reality, they usually pick a romantic comedy. Another challenge is keeping the quality of the recommendation algorithm high for those objects with very few amounts of comments. The more ratings the more input to work with and vice versa. For example, the majority of movies on Netflix have very few ratings.

It can also be difficult to measure how well the goal has been reached for some objectives. For example, if the objective is to increase user interaction, it can be hard to determine if the user just has a movie playing in the background or actively watches the movie.

References

Bell, R., & Koren, Y. (2007). Lessons from the netflix prize challenge. *SIGKDD Explorations*, *9*, 75–79. https://doi.org/10.1145/1345448.1345465

Bennett, J., & Lanning, S. (2007). The netflix prize. https://api.semanticscholar.org/CorpusID:9528522

A Code for task 2

```
import pandas as pd
import numpy as np

#Reding the data
movies_df = pd.read_csv("movie_genres.csv")
users_df = pd.read_csv("user_reviews.csv")

def set_up_dataframes(users_df, movies_df):
    # Convert DataFrames to NumPy arrays and drop unwanted columns
    genres_matrix = movies_df.drop([movies_df.columns[0],
    movies_df.columns[1]], axis = 1).to_numpy()
    #want to drop the indexing column and the user column to only have the
    #movies with ratings
    user_matrix = users_df.drop([users_df.columns[0], users_df.columns[1]],
    axis=1).to_numpy()

    #matrix multiplication of the genres and movie ratings,
    # this will give us a matrix of users' ratings for each genre
    #If row 1 has a 3 in column 1, that means that user 1 has a total
    # rating of 3 for movies of genre 1
    return np.matmul(user_matrix, genres_matrix)
```

```
#helper function to get the recommended movie
def get_recommended_movie(user_ind, movie_indexes, user_movies):
    recommend_movies = []
    for inx in movie_indexes:
        if user_movies.iloc[user_ind, inx] == 0.0:
            recommend_movies.append(movies_df.loc[inx, "movie_title"])
        if len(recommend_movies) == 5 :
            return recommend_movies
    return "No movie found"




def recommend_movies(users, users_df, movies_df):
    #This gives a list of movies with ratings for the chosen users,
    # to use later to check if a movie has been rated or not
    user_movies = users_df.loc[users_df["User"].isin(users)]
    user_movies = user_movies.drop(columns=["User", "Unnamed: 0"])

    users_rated_genres = set_up_dataframes(users_df, movies_df)

    fave_genres = []
    movies_to_recommend = []

    for user_index in range(len(users)):
        #take out the row of the user at user_index
        rated_genres = users_rated_genres[user_index]

        #This appends an array of the top three genres for user at user_index
        fave_genres.append(np.argpartition(rated_genres, -3)[-3:])

        #This gets the indexes of all movies that have all three of the
        # user's favorite genres
        idx_of_movies_with_genres = movies_df.loc[
        (movies_df.iloc[:, fave_genres[user_index][0]+2] != 0)
                & (movies_df.iloc[:, fave_genres[user_index][1]+2] != 0)
                & (movies_df.iloc[:, fave_genres[user_index][2]+2] != 0)].index
        #this appends five recommended movies for the user at user_index
        # movies_to_recommend.append(get_recommended_movie(user_index,
        idx_of_movies_with_genres, user_movies))

    return movies_to_recommend, fave_genres



#the users to recommend movies to
users = ["Vincent", "Edgar", "Addilyn", "Marlee", "Javier"]
movies_to_recommend, fave_genres = recommend_movies(users, users_df, movies_df)

#print the results
for i in range(len(users)):
    liked_genres_str = ', '.join(movies_df.columns[2 + index]
    for index in fave_genres[i])
    liked_movies_str = ', '.join(movies_to_recommend[i])
    print(f"For {users[i]}, we recommend:
    \n {liked_movies_str} \n Since {users[i]} likes: {liked_genres_str} \n")
```