CS410 Final Project Progress Report (Team Early Offer)

Qiyang Chen qiyangc2
Yanhao Qiu yanhaoq2
Xipeng Song xipengs2
Yufei Ouyang yunfeio2
Xinyi Ai xinyia2

1. Introduction

We plan to build a recommender system similar to Netflix's to recommend movies to users. It is essential and exciting because films are very close to our daily life and the performance of the recommended system will affect the users' experience while getting recommended films from the system. In addition, Netflix introduced Autoencoder into the recommendation domain, which achieved a good performance in this domain.

We plan to implement three architectures in our recommendation system, the **autoencoder**, **content-based filter** and **user-based filter** based on the public dataset on Kaggle. We have completed the investigation of those techniques by reading research papers and found relevant Kaggle datasets that can be used in our project.

2. Completed Tasks

We have completed the first three tasks in our proposal, which are shown below.

2.1 Literature Review

Autorec: Autoencoders meet collaborative filtering [1]

This essay proposes AutoRec, a new Collective Filter (CF) model based on the autoencoder paradigm; our interest in this paradigm stems from the recent successes of (deep) neural network models for vision and speech tasks. The authors argue that AutoRec has representational and computational advantages over existing neural approaches to CF, and demonstrate empirically that it outperforms the current state-of-the-art methods.

Collaborative denoising auto-encoders for top-n recommender systems [2]
In this paper, the researchers present a new model-based collaborative filtering (CF) method for top-N recommendation called Collaborative. Denoising Auto-Encoder (CDAE). CDAE assumes that whatever user-item interactions are observed are a corrupted version of the user's full preference set. The model learns latent representations of corrupted user-item preferences that can best reconstruct the full input.1 This also means, during training, the authors feed the model a subset of a user's item set and train the model to recover the whole item set; at prediction time, the model recommends new items to the user given the existing preference set as input. Training on corrupted data effectively recovers co-preference patterns. This essay shows that this is an effective approach for collaborative filtering.

A Practical Introduction to Information Retrieval and Text Mining.
Chapters 10 - Section 10.4, Chapters 11 **[3]**
We have learnt how machine learning can be used to combine multiple scoring factors to optimize the ranking of documents in web search and learn techniques used in recommender systems (also called filtering systems), including content-based recommendation/filtering and collaborative filtering.

Deep Learning Based Recommender System [5]
In the essay, the researchers provided an extensive review of the most notable works to date on deep learning-based recommender systems. They proposed a two-dimensional classification scheme for organizing and clustering existing publications. They also conduct a brief statistical analysis of these works to identify the contributions and characteristics of these studies. They highlight a bunch of influential research prototypes and analyze their advantages/disadvantages and applicable scenarios. The goal of this essay is to thoroughly review the literature on the advances of deep learning-based recommender systems. It provides a panorama with which readers can quickly understand and step into the field of deep learning-based recommendation.

2.2 Data Preparation
We searched online on different platforms to find the best movie dataset for our recommendation system. After the comparison, we find the "TMDB 5000 Movie Dataset" from Kaggle [4], which is the most suitable one for the project.

In this dataset, there are about 5000 movie rows and many useful columns from two CSV files so that we could set up our recommender system based on various content. For instance, the famous movie Avatar has the genres of action, adventure, and fantasy, the language of English, the production company Twentieth Century Fox Film Corporation, and a popular value from the first file. We could also retrieve the cast and crew from the second file. These attributes could be used to increase the performance of our recommender system.

Moreover, there are some columns such as website and overview for each movie, which could give us more choices for the implementation of the recommender website.

2.3 Data Processing

We have two CSV files for the dataset:

Tmdb_5000_credits.csv

There are 4 columns below,
columns: movie_id, title, cast, crew.
movie_id: the identity of the movie
title: the name of the movie
cast: include “cast id”, “character”, “credit_id”, “gender”, “credit_id”, "gender", "id", "name", "order".
crew: include “credit_id”, “department”, “gender”, “job”, “name”.

tmdb_5000_moveis.csv

There are 20 columns below,
Columns: budget, genres, homepage, id, keywords, original_language, original_title, overview, popularity, production_companies, production_countries, release_date, revenue, runtimes, poken_languages, status, tagline, title, vote_average, vote_count.

budget: it is in US dollars.
genres: include “id” and “name”(type of movie).
homepage: the website of the movie.
id: id number
keyword: include “id” and “name”(keyword).
original_language: language of movie English(en), (fr), (es), (zh), (de).

original_title: the movie name.
overview: description of the movie.
Popularity: official data to measure how popular the movie is.
Production_companies:include “name” (production company) and “id”(id of company).
production_countries: include “iso_3166_1” (abbreviation of country), “name”(name of country).
Release_date: movie release date.
revenue: revenue of the movie
runtimes: runtime in minutes
spoken_languages:include “iso_639_1” (abbreviation of language), “name”(name of language).
Status: status of the movie
tagline: tagline of the movie
title: title of the movie
vote_average: vote average of the movie
Vote_count: vote count of the movie

3. Pending Tasks

Completed

1. Reading related papers (20 hours)
2. Data acquisition (2 hours)
3. Data processing (20 hours)

Pending

4. Developing Autoencoder based recommended system (20 hours)
5. Developing Content-based Filter recommended system (20 hours)
6. Developing User-based Filter based recommended system (20 hours)
7. Performance analysis (10 hours)
8. Frontend UI-based Python GUI (20 hours)
9. Evaluation (10 hours)
10. Writing a final report (10 hours)

4. Challenges

1. **Different languages.** Considering Netflix is widely used in the world, and the users are from different countries using different languages. To better recommend the films to users, this presents a challenge as the

recommend system might anlyze the multiple languages. In this project, our group only focus on users who use English.

2. **Lack of data analytics capability.** Like all AI-based technologies, recommendation engines rely on data – if the system is developed without high-quality data, or cannot crunch and analyze it properly, the system will not be able to make the most of the recommendation engine. Deep learning-based recommendation engines can demand high computational complexity. If the data that is fed to the model is less accurate or valuable, the result will be less useful.
3. **The 'cold start' problem.** Relying on user data has its downsides, one of which is the issue of 'cold start'. This is when a new user enters the system or new items are added to the catalogue, and therefore, it will be difficult for the algorithm to predict the taste or preferences of the new user, or the rating of the new items, leading to less accurate recommendations. However, a deep learning model is able to optimize the correlations between the customer and the product by analyzing the context of product and user details like product descriptions, images, and user behaviors. It can then make meaningful recommendations for each individual product or customer under different scenarios. This results in a unique set of recommendations that takes into consideration all these variables.
4. **Inability to capture changes in user behavior.** Consumers do not stand still – they are constantly behaving and evolving both as people and customers. Staying on top of these changes is a constant battle. A strong recommendation engine will be able to identify changes (or signs of an impending changes) in customers' preferences and behavior, and constantly auto-train themselves in real time in order to serve relevant recommendations.
5. **Privacy concerns.** The more the algorithm knows about the customer, the more accurate its recommendations will be. However, many customers are hesitant to hand over personal information, especially given several high-profile cases of customer data leaks in recent years. However, without this customer data, the recommendation engine cannot function effectively.
6. **Efficient Evaluation.** The recommendation output should be evaluated manually. Considering the size of our group, we could not generate too

many recommendation outputs. The performance of the evaluation will affect the recommendation system final performance.

References

[1] Sedhain, Suvash, et al. Autorec: Autoencoders meet collaborative filtering. *Proceedings of the 24th International Conference on World Wide Web. ACM, 2015*

[2] Wu, Yao, et al. Collaborative denoising auto-encoders for top-n recommender systems. *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 2016*

[3] C. Zhai and S. Massung. Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining, *ACM Book Series, Morgan & Claypool Publishers, 2016*

[4] Kaggle TMDB 5000 Movie Dataset. https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?select=tmdb_5000_credits.csv

[5]Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv. 52, 1, Article 5 (January 2020), 38 pages. https://doi.org/10.1145/3285029*