CS410: Text Information System Project Proposal

1. Team ACT information

Member Name	Contact Details
Hyeonjae Cho (Captain)	hc53@illinois.edu
Xi Chen	xi12@illinois.edu
Sean Kan	clkan2@illinois.edu
Danwoo Park (NetID: danwoop2)	danwoop2@illinois.edu
Kihyuk Song	kihyuks2@illinois.edu

2. Project Description

Our team is planning to build a movie recommendation system that gets input on any queries and returns a list of relevant films, along with their corresponding keywords and taglines. The user can then go through the results, pick films that spark their interest, and get a list of recommendations that are similar to what they like.

The movie recommendation system will be built using publicly available information from the following databases:

- MovieLens 25M datasets for movie titles, genres, tags, and user ratings
    - https://grouplens.org/datasets/movielens/25m/
- IMDB for cast and director names
    - https://www.imdb.com/interfaces/
- WikiData for creating a knowledge graph that describes real-world entities and captures the relationships between them.
    - https://query.wikidata.org/

The reason why our team chose a recommendation system is that as the lecture says, future intelligent information systems are expected to be highly personalized and alleviate users' effort to perform a task. The recommendation system is the way the information system desires to be. In addition, this project is important in that we are trying to utilize the WikiData Knowledge Graph algorithm, which is a state-of-the-art recommendation algorithm.

We will build the BM25 model first which is used to retrieve movies relevant to the query. At the same time, we need to construct the knowledge graph with WikiData API extracting movies-related real-world entities and relationships. Since we have user rating data, we would like to develop a collaborative filtering model, which would recommend content purely based on the rating data. In the end, the hybrid model can be summarized as below:

Recommender = λ * Results from the collaborative filtering model + (1- λ) * Results from KG created by the WikiData API. Where λ is a parameter that we can control (λ = [0,1]).

We expect that each model will produce different recommendations. The KG model can recommend based on the topology it built from real-world entities so it can capture more details such as people, places, and things. On the other hand, collaborative filtering is based on user rating data which represent preferences that the KG model cannot handle.

The output would be a simple web page that gets movie title inputs from users. The page would return search results that show related contents and entities inferred from the model.

Evaluation of the recommendation system

MovieLens 25M dataset has 162,000 users' ratings.
For each user, we can split the dataset into three kinds, which are training, validation, and testing. We will not use the test dataset to enhance our system to avoid look-ahead bias, so we should split the dataset into three, which includes the validation dataset.
Since ratings are made on a 5-star scale from 0.5 to 5.0 with a half-star increment, we can evaluate the recommender system's score prediction performance by calculating the RMSE using the actual score value and the score our system predicted for the user.

Evaluation of the search system

To evaluate the search system, since we can assume that the results of the query should not be that many compared to the general search engine, we may use the F2 measure to evaluate the BM25-based movie search system as we should be more focused on the recall. We may easily build the test set from the IMDB and the MovieLens data since we can map the title, genre, and crews to the movies.

3. Language

Mainly Python. More languages can be used when building a demo webpage, which will show the search result of given inputs.

4. Breakdown

Since we have five members on the team, we expect to spend close to 100 hours on this project. Here is a list of main tasks to be completed and their estimated time cost:

Task	Estimated Time
Data Preprocessing - Data integration between MovieLense and IMDB (Movie, tag, genre, crews)	5 hrs

Text Retrieval System with BM25 - Use MeTA library to build our own BM25 model	15 hrs
Collaborative Filtering System - Based on MovieLens user rating data	15 hrs
WikiData Knowledge Graph - Use pre-built KG by WikiData API, extract movies-related entities from matched movie titles result	15 hrs
User Interface Design	10 hrs
Search Result Page - Develop a web page that gets input and returns corresponding results	15 hrs
Main Function - Integrate model outputs with webpage	5 hrs
Model Evaluation	10 hrs
Testing / Debugging	8 hrs
Demo Creation - Take a video that shows how our search result page and codes run behind	2 hrs
Total	100 hrs