Project Proposal

Fresh Tomatoes

Chaeeun Moon, Jaehyun Bhang,
Kyeonghwan Gwak, Yewan Na

Professor Mijung Kim

Software Engineering (CSE 364)

22 March 2024

Fresh Tomatoes

Chaeeun Moon Jaehyun Bhang Kyeonghwan Gwak Yewan Na

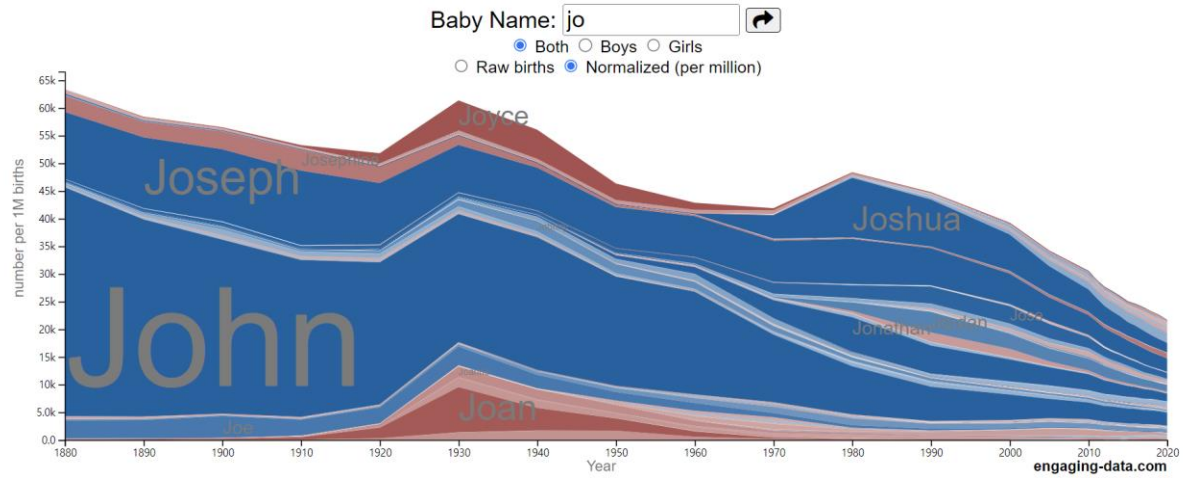
1. Introduction

We often spend a lot of time and feel inconvenience searching for movies whose content we vaguely remember but whose exact names we cannot recall, searching for movies that were popular in different eras, and searching for worthwhile movies to watch. Hence, we would like to develop **Fresh Tomatoes**, a web application that provides following three key services with movie data: LLM based movie search engine, movie popularity visualizer, and user centred movie recommendation system. As a result, users will be able to search for the movies they want to watch more efficiently.

2. Key Product Features

First, LLM based movie search engine. We sometimes want to rewatch some movies we've already seen. We often search for movies that we have been recommended from various sources such as YouTube, TV, friends, news, etc. In that case, we frequently spend a lot of time searching for the title of that movie. And we sometimes cannot find the movie title, even at the end. Here, we would like to introduce a LLM based movie search engine. We will develop a service that uses LLM to take inputs such as a roughly remembered plot, genre, actors, background music, etc., and outputs possible candidates for the movie user is looking for. For this, we will basically use the ChatGPT API and fine-tune the LLM so that it works well for this specific task and some challenging cases.

Second, movie popularity visualizer. Some users want to get some historical views on the movies, such as movie popularity over time. And that can guide the users for the next movie they may watch. Hence, we would like to introduce a fascinating visualization of the change in movie popularity over time. We are going to use timestamps attribute in the Ratings data to represent the time when reviewers saw the movie. Since the timestamps are distributed unequally, we will categorize them into each year. And the counts of review timestamps of the movie will represent the popularity of the movie. So, we have to derive new dataset from the original one, which is, counts of timestamps of each movie over time(year). Similarly, we can also derive new data such as, counts of timestamps of each movie category over time(year). By using this data, visualizer takes a movie title or movie category as an input and visualizes the popularity of those movies over time. We set the below visualizer as an example.



< US Baby Name Popularity Visualizer >

Third, user centred movie recommendation system. When we have some free time and decide to watch some movie, we spend hard time to find worthwhile movies to watch and really need something that provides many tempting recommendations. Hence, we would like to develop a user centred movie recommendation system. It takes information such as the user's age, gender, and occupation, and based on the movie data that similar people liked, it provides recommendations that suit the user's taste. We can achieve this by first partitioning the set of rating users into categories and specify the category to which the current user belongs. Then, sort the movies so that those with a higher average rating given by users in the category appear with higher priority.

3. Concrete Project Plan

	Feature 1	**Feature 2**	**Feature 3**
Before the deadline of milestone 2	Implement the most basic ChatGPT API in our web application. So that users can use LLM within our web services.	Derive new dataset to support the movie popularity visualizer	1. Implement the user login system that requires the user information at the time of registration. 2. Implement the partition of user ratings based upon the user information (Gender, Age, and Occupation; 2*7*21 buckets) 3. Implement the sorting algorithm and display e.g. top 5 movie ID or name of the bucket of current user.
Before the deadline of milestone 3	1. Create multiple test cases using humans and external LLM. 2. If our LLM fails to pass a test case, fine-tune it appropriately to pass the test case. 3. Repeat the above process to continuously improve the performance.	Front-end 1. (default) visualizing stacked area chart (representing whole movies) 2. (user input; movie title or movie category) Visualizing area chart of the input movie 3. normalization function for each visualization	1. Implement the front-end of login and recommendation feature finally. 2. Test the implementations and revise if necessary.