CS410 Project Progress Report – Team DJTV

Team Members

Danh Nguyen NetID: danhn2@illinois.edu

Justin Quach NetID: jmquach2@illinois.edu

Tik On (Johnson) Lui NetID: tlui2@illinois.edu

Vamshidhar Kommineni NetID: komminen@illinois.edu

Task Updates

We built an initial task list in our proposal. Below is the same task list (excludes the stretch goals since we haven't gotten to them yet) with new **columns describing the completed and pending task status as required**: *Completion Status, Actual time spent and Comments/Notes*

Task Type	Task	Description	Completion status (as of 11/12)	Actual time spent	Time estimate in hours as of 10/23 (initial, low confidence)	Comments/Notes
Research	Data Review	Understanding the data structure of the primary movie dataset	Completed	4	4	We each understand the research data set which contains 100,000 user reviews
Research	Package Review	Read and understand the documentation to understand how to use Cinemagoer API	Completed	4	4	We reviewed the API and confirmed that it can be used for our application to get metadata from IMDB
Design	Webpage Design	Designing a simple web application	In progress	2	4	We are in the process of designing the simple web application that will host the search engine and other information system features
Implementation	Fetch and store metadata	Fetch all structured metadata from IMDB matching the movie dataset	Completed	10	2	We have used the API to download structured metadata for all data. More time spent for data validation than actual creation of the code for download
Implementation	Structured data storage	Build a file-based system (e.g. simple	Complete	8	8	All metadata related to the movies from IMDB and the

		JSON files) to store the retrieved metadata				research dataset is stored as JSON files. We may update this format over time as we implement the features.
Implementation	Web Application implementation	Implement the front end for different scenarios – Browse, search, etc.	Pending		16	Will implement this in the remaining weeks
Implementation	Movie Search Engine – **Uses CS410 learning**	Implement a movie search engine by using user's queries to get results from the database	In progress	10	16	Basic prototype using MetaPy using movie names as corpus. Also started looking at a Flask backend to receive queries and return results to the user.
Implementation	Sentiment Analysis – **Uses CS410 learning**	Implementing a sentiment analysis classifier to detect positive or negative reviews	In progress	9	16	Working with a Jupyter notebook to try sentiment analysis and use the dataset labels for training data
Implementation	Similar Movie Feature – **Uses CS410 learning**	Implementing a Similar Movie Finder to get similar movies in movie detail page	In progress	8	16	Implemented 3 methods for similar movies - random walk, Jaccard, TFIDF. Work ongoing to test performance (Mean DCG@5)
E2E scenario testing	Integration and scenario testing	Writing wrap-up scripts to run and bundle all the required scripts and test the feature set	Pending		16	Pending
Documentation	Application Documentation	Writing a user-friendly readme.md to use and test our application per grading guidelines	Pending		2	Pending
Documentation	Presentation	Preparing a storyline and recording(s) a video to demonstrate our works	Pending		4	Pending
Documentation	Final Report	Final report with details on design, implementation, etc.	Pending		8	Pending
Total				**55**	**116**	

Challenges

- Movie search engine

    - We had to make several design decisions on the data transformations. For now, we have several .dat files that store all the results of each data transformation step.
    - We needed to experiment with parameter tuning in metapy to get the most accurate results
    - Running the search engine will require a web backend to host the metapy based python module and receive/respond to search requests from the Javascript in the client-side web application. We're still looking into the best way to do this
- Sentiment analysis of reviews
    - Since invoking the 'get_movie' function only returns 25 movie reviews, there is a chance that returning the f1_score would not work because most, if not all the reviews, will share the same sentiment analysis score. The way we calculated the sentiment score was to return the sentiment label given that label's corresponding max value. For example, with 25 movies, all of the scores from the sentiment analysis function could be neutral, and the f1_score function will fail without two or more sentiment score labels.
- Similar movie feature
    - We found that the performance of TFIDF is worse than Jaccard distance, we will need to investigate this further to figure out the root cause
- Web application
    - Since none of the team members are that familiar with web applications, we have been looking into the different web application creation methods and seeking to maximize use of client-side JS to the extent possible. We'll make progress on this over the coming few weeks including the server side implementation needed for the search engine.

Details on Tasks

Below, we include a bit more detail on the core tasks relating to the project:

Fetching and Storing metadata and reviews

We leveraged the Cinemagoer API to fetch the metadata that we required:

- localized_title: Movie Title
- cast: Casting of the movie
- genres: Genres of the movie
- runtimes: Total runtime of the movie
- rating: Rating of the movie
- year: Release year of the movie
- producer: Producer of the movie
- director: Director of the movie
- cover_url: URL of the movie's cover
- summary: Short Summary of the movie

We stored this information along with the corresponding reviews from the research dataset into compressed json format. We implemented compress and decompress functions for the modules which need the Json data.

Movie Search Engine

We built a movie search engine with MetaPy with the movies names as the corpus. This basic movie search engine works and yields relevant, ranked results. We are looking at adding more features like search by year, actor, etc. We are also looking at adding support for different rankers and allowing the users (CS410 users are the target demographic) to choose the ranker to use.

Sentiment Analysis of Reviews

This is working in a Jupyter notebook and returns the sentiment score for the labels (negative, neutral, positive, and compound [overall score]) for each individual review. We have more experimentation and work to do here including incorporation into a final form that is usable by the web application.

Similar Movies Feature

The Vocabulary Building Function has been implemented and it includes

- Changing all the words into lower case
- Special Characters and Stopword removal to remove non-meaningful input
- Word Tokenization to minimizes text ambiguity

The Performance evaluator has also been implemented to have a standardized evaluation. As we don't have label data to determine whether the ranking is correct or not, we leverage the genres to compare the target movie and similar suggested movies. If they have common genres, we will define them as being similar. DCG@5 is used as our metrics

Methodologies used:

- Randomwalk
    - This is used as a baseline comparison, we just sample 5 movies from the pool without replacement as the similar movies result.
- Jaccard
- TFIDF