CS 194 Proposal: Watch With Us

Jason Block, Palani Eswaran, Alison Kohl

Watch With Us will be a movie recommendation engine that takes the preferences of a group of people into account to recommend a movie for the group to watch together. It will be a web application optimized for mobile.

In brief: The first interaction a user has with the app will be rating a number of different movies in order to learn his or her tastes. From there, the app will suggest a number of movies based on the user(s)' data. The users can then rate the movies after watching them to improve their future recommendations.

In-Depth Description: When a user first opens the application he or she will be asked to rate a number of movies on the Watch With Us scale, which consists of four options (shown below). An image of the movie poster will be displayed, followed by the four options and corresponding icons, and a separate "Don't Rate" button. Clicking on the movie poster will bring up more detailed information, including the Rotten Tomatoes and IMDB ratings and a short summary for the given movie. The APIs will be queried for this information on a movie-to-movie basis.

The rating options will be two thumbs down, one thumb down, one thumb up, and two thumbs up. Each will be associated with an appropriate icon. This will simplify the rating process for users and create more consistency amongst ratings between different users, since users may have different opinions of what a given numerical rating represents. The movies initially suggested upon first opening the app will be from varying time periods and will be what we consider to be representative of their respective genres. Users will be required to rate at least ten movies to help us get a baseline sense of their tastes, but the user can rate as many movies as they'd like. After entering these initial ratings, the application will begin to become aware of the user's specific tastes. The user can also bring up the initial rating system at any other time to edit their past ratings or rate new movies to make their preferences more personalized.

When a user is ready to watch a movie, the application will scour a number of different movie databases, including IMDB, Rotten Tomatoes, and Metacritic, and will make a number of suggestions based on the user's preferences. We will use linear regression to correlate the user's personal ratings with both the average user ratings and data from individual critics that is available on these APIs. The specifics of these linear regressions are described in the "Potential Approaches" section.

A user will be able to indicate if they are watching a movie with a group of friends. If so, they will select which friends, and those users' preferences and ratings will be taken into consideration. The engine will attempt to suggest films that are believed to be a good balance of all of the users' preferences.

While receiving suggestions, the user will see a single movie poster displayed on the screen at any given time (similar appearance to the initial rating process). Users will be able to swipe upwards to rate a given movie at any time with the same rating options described above if they've seen the movie before or know they would never want to watch it. The user will be able

to swipe downwards on a movie poster to flag a movie to “compare” to other flagged movies they may be interested in watching. Swiping left and right will scroll horizontally through the movie suggestions. These flagged movies would be contained in a separate list accessible at any time within the application.

After selecting a movie that the individual/group would like to watch, the application will send a notification to all users prompting them for their rating, after watching the movie. This rating will affect the suggestions of the given user in the future. We will use several algorithmic techniques to provide suggestions from the user, utilizing multiple movie data APIs.

Need for the Product

This product is especially useful for college students and other young people who often gather in groups to watch films. Speaking from our own experience, it can be very difficult to find a movie that no one in a group has seen and is pleasing to everyone watching. The ability for the application to access a number of different movie databases is also useful because it ensures that the user can select from the widest array of options.

It is also very useful to be able to record feedback on a film immediately after one sees it and to then use that feedback to influence future habits. It is very easy to forget one’s feelings about a film after going a long time without seeing it.

Potential Audience

We already mentioned a large portion of our target audience - college students who want to watch a movie together. However, the audience can really be any group of people who want to spend time together. Families would greatly benefit from this product as it is even harder for multiple people of different ages with different ages to agree on anything. It is also difficult to find movies that are age-appropriate for families, so we may take MPAA ratings into account. Couples are also a huge potential audience. The app will be able to function as a movie recommendation engine for an individual.

Discussion of Competing Products

The biggest competing products are existing movie streaming sites such as Netflix and Amazon Instant Video. However, the primary feature that these products lack is the ability to look at movies that are not present on their sites. These sites also do not take group watching into account. Netflix, in particular, separates a shared account into multiple profiles, which can cause conflicts amongst a group.

Other competing products are movie rating sites such as Rotten Tomatoes, IMDb, and Metacritic. These sites, however, only take one rating system into account whereas ours would leverage multiple. And once again, these sites do not use personal or group preferences into account when recommending movies.

Major Technologies Used

We will build this application in Node.js with the Express framework. We are familiar with Node.js and Express from CS147 and CS247, and we feel that this is a good framework for a consumer-facing application. The application will be mobile-friendly and thus users can use it easily on a computer or a phone. We plan to use Bootstrap to make it optimized for mobile but usable on a variety of platforms.

Our database will store user data - their demographic, their past movie ratings, and other personal preferences. We plan to use Firebase, as it is quick to set up and supports user authentication, a useful feature that will save us a fair amount of work.

We will also utilize technologies that make mockup creation more efficient. We have experience with both Omnigraffle and InVision, and will use these to create wireframes of our interface to gather opinions from users.

After we build our first prototype, we'll want to do user testing. A lot of this will be done in person, but we will also use Google Analytics for A/B testing.

We will also be leveraging multiple APIs to find movie information and initial ratings. Some of these APIs include OMDb and (Open Movie Database) and the Rotten Tomatoes API.

We will host our application on Heroku during development, with the possibility of moving to a different domain when the application is in a more complete state.

Resource Requirements

We don't expect to need too many resources outside of our database management system and the APIs mentioned above. As we continue to develop our application we may realize that we need to utilize paid APIs, such as the IMDb API, to strengthen our data.

We are also considering buying a domain name when the project is more complete. We will revisit this idea in the final weeks of the quarter.

Potential Approaches

We will approach building this application by first figuring out how to leverage the APIs and store proper information in our database. We will then need to figure out an algorithm that takes in this information and calculates a number of suggested movies.

We will use a linear regression approach to predict the movie rating estimate for users. Using the ratings the users have already given to movies, we will perform multiple linear regressions between the ratings and datasets that we obtain from public movie databases, such as IMDb and Rotten Tomatoes.

In the case of IMDb, the four-option rating the user has assigned will be extrapolated to the 10 point scale used by IMDb. We will obtain weights that differ between genres, time periods, and more for each user that will be used to compare their ratings to IMDB ratings. When the

application makes new suggestions, it will suggest movies that are highly rated after being multiplied by the user's specific weights.

In the case of Rotten Tomatoes, we will perform a linear regression between the user's ratings and both the average rating for a given movie across all critics, as well as specific critics (whose ratings can all be pulled with the API) whom the user seems to have aligned ratings with. Eventually, we will ideally be able to match up the user to several critics with whom they agree, and make future suggestions of movies that those critics have rated highly.

These two will be the primary movie database APIs that we will utilize, but we will also attempt to add further data such as pulling from databases like Metacritic, which also aggregates both user and critic ratings.

Assessment of Risks

The major risk is that we are unsure exactly how our movie suggestion algorithm will be implemented. Some team members are familiar with linear regression; however, we must figure out which parameters we will be using in our regression. Some of these parameters may include genre, directors, actors, year, etc. After running user tests on our initial prototype it may become clear that our suggestions aren't accurate. If this is the case we will have to modify our algorithm to emphasize different parameters. We may need to add an API if we feel that we could use additional data or delete an API if we feel that its data is detrimental to the algorithm.

Another risk we are taking is the design of our user interface. While we do feel that the interface is minimalist and will serve the user well, there might be some hiccups that we don't encounter until users are actually interacting with the application.

A small risk is the reliability of the APIs. We have no reason to believe that these APIs will ever go offline, but in the past we have had issues with APIs crashing and/or not responding while working on a project.

Next Steps

Our first step will be setting up a Node.js application, deciding on a database management system, uploading the project to a Git repository, and deploying it to Heroku. Once these basics are completed we will begin designing our application using Bootstrap so that it is optimized as a mobile web application.

At this point, we will probably split up the work and have one person working on the user interface, one person working on handling API calls, and one person working on the linear regression algorithm. In our estimation, both the interface and API handling will be fairly straightforward, so once these tasks are completed the entire team will likely focus on the algorithm.

As soon as we complete our first working prototype we will constantly be user-testing. This will help us in two ways: improving our interface and improving our algorithm. In order to bolster our user-testing we will probably use Google Analytics to facilitate basic A/B testing. We will A/B test both on the appearance of pages and on the implementation of our algorithm.