CS51 Final Project Draft Speculation

Peter Chang (chang04@college.harvard.edu), Dor Baruch (dorbaruch@college.harvard.edu)

Zachry Bai (zachrybai@college.harvard.edu), Annie Hwang (annhwang@college.harvard.edu)

Project Title: Moooooovie Ranking Predictor Using Random Forest

---

Brief Overview:

The ratings that appear on rottentomatoes.com or imbd.com seem to be governed under a list of significant factors that contribute to the rating that is given to each production. The ratings on these websites and other ratings from critics are generally most dependent on the plot summary. The director, who is responsible for interpreting the script, and the actors, who are responsible for giving life to the characters, are also significant factors in rating a particular movie. While the ratings on IMBD, rottentomatoes, and other movie review websites are certainly helpful for helping consumers choose which movies to invest their time and money in, they are not so useful for helping consumers choose a movie to watch that has been released recently.

Our project attempts to discard this little problem of not knowing which movie-in-theater to watch. Our project predicts the rating of a movie using data of the cast members, plot summary, director, and other core factors that contribute to important aspects of a movie. We plan to use web-crawlers that links a movie review url to the links of specifically the cast list, director, plot summary, etc. We also plan on using Random Forest as our core machine learning algorithm that will use decision trees which will map the observation of the given data (cast list and the ratings of movies that they were in, past rating of films by director, etc) to conclude about the movie's rating. Our goal for the project is to predict the critical rating of a movie and minimize the mean squared error of our prediction compared to the actual rating.

Feature List:

We have a number of features we would like to implement which vary highly in time/computational complexity. We first need to implement a web-crawler that will allow us to crawl through a movie-review website (iMDb.com) and extract the basic info about movies. We will extract the first five (principal) cast members, the director, the producer, the basic plot summary, and the user rating using the web-crawler.

Using the database extracted by the web-crawler, we will implement a supervised learning algorithm to fit a regression that allows us to predict the rating of the info for a new movie based on the basic information. We are currently thinking of implementing a random forest regression as our supervised learning algorithm, as it is cited by a number of literature as among the most consistently accurate supervised learning algorithms, and also due to the nature of its implementation, it avoids over-fitting.

Technical Specification for the web-crawler:

We will build our web-crawler following the guide in

http://www-rohan.sdsu.edu/~gawron/python_for_ss/course_core/book_draft/web/urllib.html

The web-crawler will use built-in Python libraries like urllib and HTMLParser to extract the html from each URL that is visited. Then, the function will look for keywords in the html page to extract the wanted information (<h2>Storyline</h2> for storyline, <span class="itemprop"

itemprop="name"> for actors etc). Lastly, a text file will be created to which all the parsed information will be added before visiting the next link in the list.

Technical Specification for the Random Forest algorithm

Random Forests are a learning method that use a set of decision trees to come up with a regression based on the individual outputs of the decision trees. By using the aggregation (bagging) of multiple decision trees with a random selection of features, overfitting is minimized and accuracy will be increased.

First, we will need a module for each individual decision tree. The module will consist of nodes, as is typical of a tree, with a value and a left and a right branch. The value will ultimately be a query, e.g. “Is Brad Pitt in this film”, with the left branch representing “No” and the right, “Yes”. Each decision tree will question if a random subset of features are in the film, i.e. (director, producer, one particular actress). As is standard of any tree module, we will need an insert, remove, member, and last (takes the set of the values that are attached to the leaves) functions.

Using cross validation, we will decide on the number of decision trees to use for our random forest. After training, when making predictions, we will aggregate the results of the individual trees by clustering the results by the first digit of the output rating, (for example, if the rating for a particular route through the tree is a 8.4, we will place it in a cluster of other ratings whose first digit is an 8) and take a “vote”, seeing what cluster the majority of the trees vote for. Then, within that cluster, we will average the value of all the ratings for the decision trees to give the final prediction for the movie with an unknown rating.

Next Steps:

Talking with our TF to figure out exactly which machine learning algorithm would be the most feasible for our project, and exactly how to implement it. Also, we would need to create a Git repository for convenience, and agree upon which programming languages to use (python seems to be the best language right now).