# Predicting the movie popularity using user-identified tropes

**Amy Xu**
Stanford Univeristy
xuamyj@stanford.edu

**Dennis Jeong**
Stanford Univeristy
wonjeo@stanford.edu

## Abstract

Tropes are recurrent themes and narrative frames that appear throughout literature, news, and popular media. We built a model that uses character- and plot-based user-identified trope labels (source: TV Tropes) to predict audience ratings for movies (source: IMDb). First, we used the DBTropes.org and node-imdb-api APIs to collect our initial data. Then, we implemented and trained a naive Bayes model and k-means clustering algorithm as first experiments for supervised and unsupervised learning, respectively. We found that
**TODO**: briefly summarize results
**TODO**: breifly summarize analysis
Our next steps include
**TODO**: list next steps.

## 1 Introduction

The credits roll. You release your breath, maybe for the first time in a while, and lean back in your chair. *That was a* great *movie,* you find yourself thinking. It's hard to immediately articulate why, but... in general you liked the characters, the plot, the tone.

How does this happen? Or, more specifically, what makes a cinematic story feel compelling to an audience?

In this paper, we attempt to wrangle that question into an ML-friendly form and answer it using supervised and unsupervised learning. We use IMDb audience ratings data as a proxy for how compelling a movie is, and user-identified trope labels from TV Tropes to encapsulate story elements such as plot and character.

Why do we choose to look specifically at tropes? The word "trope" is defined as "a significant or recurrent theme; a motif" by the New Oxford American Dictionary. Tropes are used throughout literature (Ciarlini et al., 2009; Reagan et al., 2016), news (Boon, 2017), and popular media to express structured yet abstract concepts. In particular, they can act as "narrative frames" that are "a critical part of how we make sense of – and describe – the world around us" (Boon, 2017). Currently, there exists very little research investigating tropes from an ML perspective.

Ultimately, our goal is to

(a) build a model that uses character- and plot-based trope labels to predict audience ratings, which then allows us to

(b) evaluate whether these story elements are an effective predictor of audience enjoyment, and

(c) gain insight into how various narrative frames are perceived at a cultural level.

## 2 Related Work

In the literature, there is a precedent of building ML models to predict the success of a movie. Doshi et al. (2010) used a linear regression model with social network analysis, sentiment analysis, and movie rating features to predict movie stock prices on the Hollywood Stock Exchange. Oghina et al. (2012) used a linear regression model with statistical and textual features from Youtube and Twitter to predict movie ratings on IMDb. Our paper builds upon this prior research by introducing story elements and user-identified tropes as features.

Additionally, there are numerous papers that investigate algorithms for building personalized movie recommendation systems, as well as recommendation systems in general. Park et al. (2012) reviewed 210 articles from 46 journals published between 2001 and 2010 that were about recommender systems, 53 of which were about movie recommender systems. While this field of research

focuses on adapting recommendations to individuals, our paper takes a more aggregate view of audience enjoyment of a movie.

As for research on tropes, Reagan et al. (2016) found that most written literature can be categorized into one of six categories: "Rags to riches" (rise), "Tragedy" (fall), "Man in a hole" (fall-rise), "Icarus" (rise-fall), "Cinderella" (rise-fall-rise), and "Oedipus" (fall-rise-fall). The results were calculated using sentiment analysis of 10,000 word windows and were consistent across several algorithms, including principal component analysis, hierarchical clustering, and self-organizing maps. In our paper, the tropes are user-identified rather than organically generated, which leads to more fine-grained distinctions but also potentially more variance in trope quality.

Finally, Boon (2017) used implicitly crowd-sourced tropes from Twitter to build a system that helps readers become more aware of news misinformation. The system identifies tweets that contain a trope name and news article link, and informs the reader so that they are aware of biases in the article beforehand. This implicit crowd-sourcing strategy is especially important in the domain of news, and less applicable to movies for which there exists an explicitly crowd-sourced TV Tropes database.

## 3 Methods

### 3.1 Data and Model

TV Tropes is an online collaborative information environment, or wiki, with a collection of over 60,000 works linked to over 20,000 tropes. The site can be categorized into Trope pages, which list by category (ie. Film, Literature, Video Games, etc) the works associated with a trope, and Work pages, which list associated tropes in alphabetical order. To access trope data, we used a linked data wrapper for TV Tropes called **DBTropes.org**[1].

IMDb is an online database of movies, TV shows, and video games, containing over 4.6 million titles. Users can submit edits to pages and rate titles on a scale of 1 to 10. The edits are approved by the site, and ratings are converted using a weighted-mean average.

Additionally, Rotten Tomatoes aggregates critical reviews, and assigns them a freshness rating, which is the percentage of critics that had a positive reaction to the movie. If the movie has a freshness rating of greater than $60\%$, then the movie is considered "fresh", otherwise it is considered "rotten". To access movie rating data, we used a non-scraping api called **OMDb api**[2].

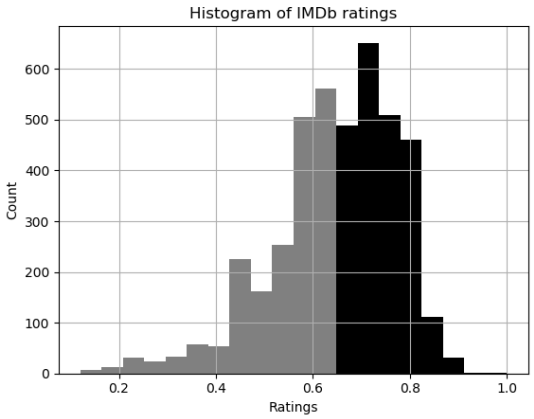


Figure 1: Histogram of IMDb ratings. The black elements are those with values above the median, and the grey elements are those with values below the median.

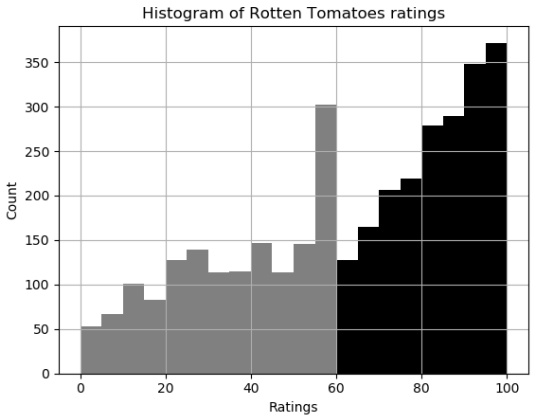


Figure 2: Histogram of Rotten Tomatoes ratings. The black elements are those with values above the median, and the grey elements are those with values below the median.

The structure of TVTropes can be modeled as a directed, bipartite graph, with each node being either a Trope or a Work, and edges being links between a Trope and a Work. Trope pages will contain a list of Work pages as examples of embodying the trope, while Work pages contain a list of Tropes that the Work contains. Additionally, edge also contains a short description of how the Trope appears in the Trope.

[1] http://skipforward.opendfki.de/wiki/DBTropes

[2] http://www.omdbapi.com/

To generate our initial dataset, we began at the Plots[3] page and followed the links to each Trope page listed. From there, we followed the links to each Work page that was also listed under the Film category. Using these work pages, we generated dataset tuples: $(\texttt{work\_name}, [\texttt{trope}_1, \texttt{trope}_2, ...])$. Then, if necessary, we queried IMDb to generate label tuples: $(\texttt{work\_name}, \texttt{imdb\_rating})$. Separately, we kept a map $\{\texttt{trope} : \texttt{trope\_text}\}$ for later experiments.

### 3.2 Algorithms

**Naive Bayes**: First, we used a sk-learn's Naive Bayes classifier with Laplace smoothing to establish a baseline for supervised learning. To implement the classifier, we used the bag of words model to create feature vectors for each movie, treating each link from a plot trope to a movie page on TV Tropes as a single word. We tried using the tf-idf model, but we ran into issues getting it to work on certain datasets. For the IMDb ratings, our labels will be 2 buckets, one for a score above the median score, and one for a bucket below the median score. For Rotten Tomatoes, we used the Rotten Tomatoes cutoff of $60\%$ for being "fresh" or "rotten" as our labels. We chose to use binary classification instead of linear regression since numerical rankings create arbitrary granularity within a measure that is inherently subjective.

**Logistic Regression**: We also used a logistic regression model to perform classification. We had an identical setup with the Naive Bayes classifier, with the addition of L2-regularization to try to avoid overfitting. This model was chosen to create a comparison against the performance of Naive Bayes.

**$k$-Means Clustering**: Next, we implemented and trained a $k$-means clustering algorithm to see if any insights could be gained through unsupervised learning. Our feature vector was identical to the Naive Bayes classifier. We tested values of $k$ in the range of 2 through 10 (the maximum number of clusters we could reasonably work with).

[3] http://tvtropes.org/pmwiki/pmwiki.php/Main/Plots

### 3.3 Feature Experiments

Our experiment experimented with three sets of features:

**Plot tropes**: All tropes present in the Plots page; we know that these specifically relate to the plot of the movie. However, there are only approximately $500$ of these, and our metrics showed that the median movie only had 2 tropes.

**All tropes**: This is the total list of tropes that are linked to the movie node. The median number of tropes linked to a movie in our dataset is $647$, which is significantly larger than the number of plot tropes.

**Trope description text**: When a trope is listed under a specific movie, it is given some descriptive text further explaining how the trope applies to that movie. For example, for the trope "Crossing the Desert" and the movie "The Ten Commandments", the user-generated description reads: "Moses does this after he's thrown out of Egypt. Ramses only gives him one day's rations, figuring he'll die that way, but he manages to make it to Midian." The median number of words of trope description text is $4517$.

## 4 Results

First, we will discuss our results from our binary classification algorithms:

### 4.1 Only Plot Tropes

For the milestone, we used only Naive Bayes with only the plot tropes as features; this did very poorly on both datasets. In a test set of 1491 movies, 709, or $47.5\%$, were incorrectly classified, which is the same performance as chance. Likewise, classifying movies as rotten or fresh according to Rotten Tomatoes did very poorly; out of $1172$ movies, $639$, or $54.5\%$, were incorrectly classified.

### 4.2 All Tropes

Once we added all tropes after the milestone, our accuracy increased. Both algorithms achieved accuracy around $65\%$ for both labeling IMDb scores and Rotten Tomatoes scores.

### 4.3 All Tropes and Comments

Adding the comments to the feature vector seemed to decrease the accuracy of the predictions across the board, except for Logistic Regression using the IMDb labels. This might be because the IMDb

| | precis. | recall | accur. |
|---|---|---|---|
| **Naive Bayes** | | | |
| IMDb | 0.65 | 0.65 | 0.64 |
| Rotten Tomatoes | 0.79 | 0.67 | 0.65 |
| **Log. Regression** | | | |
| IMDb | 0.57 | 0.67 | 0.64 |
| Rotten Tomatoes | 0.74 | 0.67 | 0.64 |

Table 1: Test results using only trope features for naive Bayes and logistic regression models.

data fits better with the underlying assumptions of logistic regression, which assumes the data is distributed as a Bernoulli variable.

| | precis. | recall | accur. |
|---|---|---|---|
| **Naive Bayes** | | | |
| IMDb | 0.61 | 0.63 | 0.62 |
| Rotten Tomatoes | 0.77 | 0.65 | 0.63 |
| **Log. Regression** | | | |
| IMDb | 0.60 | 0.66 | 0.64 |
| Rotten Tomatoes | 0.70 | 0.64 | 0.61 |

Table 2: Test results using both trope and comment features for naive Bayes and logistic regression models.

### 4.4 k-Means Clustering

k-Means clustering seems to work to a limited degree. When set to $k = 7$, manual inspection of the clusters showed some coherence. One cluster was comprised mostly of action movies, such as super hero movies (All 3 Iron Man movies, The Dark Knight movies, Thor), spy movies (Goldfinger, Die Another Day), and other movies (Indiana Jones, Inspector Gadget, RoboCop). Another cluster was comprised of sports movies and detective films. Below is a graph depicting silhouette scores for clusters when $k = 7$.

## 5 Analysis/Findings

When going from just the plot tropes to all other tropes, our accuracy improved to being better than chance. The diagnostics below of errors plotted against increasing training set size show that our training error increases as we increase training set size, which indicates that our model is overfitting, despite us using L2 regularization and normalizing the input vectors. The test error seems to stay

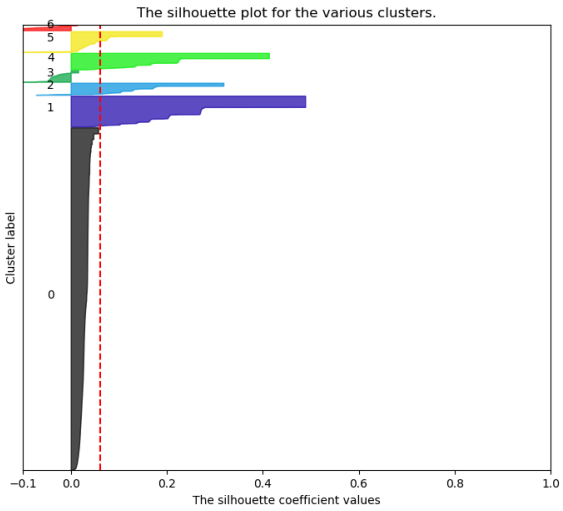


Figure 3: Silhouette score diagram of optimal k-means results. These clusters roughly correspond with the categories of drama, mystery, fantasy, and action/horror.

constant throughout, even as we add more training data. This suggests that either our model is incorrect, or more features are required.

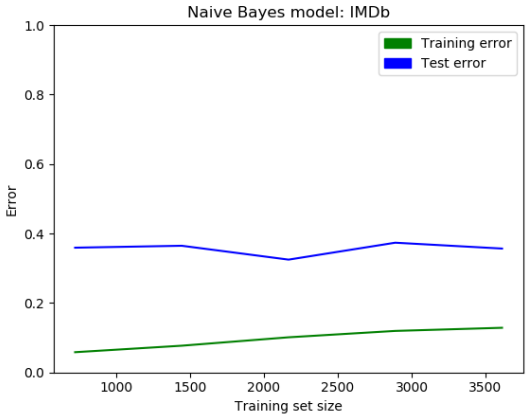


Figure 4: Training and test error of Naive Bayes on classifying IMDb scores.

We tried applying tf-idf weighting to the data in hopes to correct against the negative impact of the comments on accuracy, but that further decreased our performance. This is likely because the text of the comments has high variance, even when accounting for stop words.

For the k-Means clustering, we did not have a way to effectively empirically evaluate the clusterings. This meant that we could not draw meaningful conclusions from those experiments.

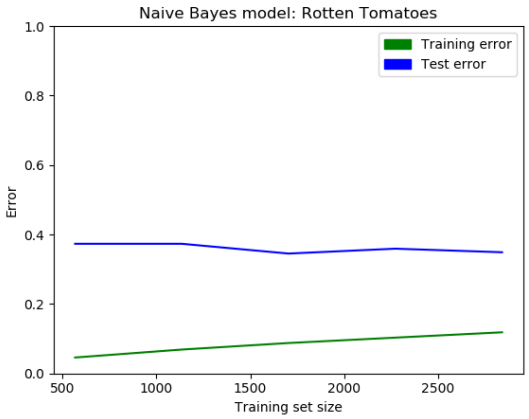


Figure 5: Training and test error of Naive Bayes on classifying Rotten Tomatoes scores.

## 6 Conclusions/Future Work

Our results seems to imply that the narrative tropes that are used to create a movie are not very predictive of the final quality of the movie. This makes some intuitive sense; two movies may share the same story beats, but be vastly different in quality of execution. We have two plausible explanations for why we do see an increase in accuracy once we add all tropes as features:

1. Tropes capture the zeitgeist of the moment, which loosely translates into higher scores for movies that can ride that trend.

2. Using the same tropes can match movies with the same directors or writers, which can be a predictor for quality (bad writers generally make bad movies, while good writers generally write good movies).

If we had to redo our project, we would want to look at even more movies; we were very limited by our data scraper because we could not find a way to query the RDF database directly.

We think that the most promising direction moving forward would be to embrace the trope's usage as narrative shorthand; just as generative algorithms can help authors come up with interesting sentences as they write[4], what if it could also help suggest high-level plot structure?

## 7 Contributions

Both team members collaborated on both the coding and writing parts of the project. For the code, we did pair programming in person. For the writeup, we used a ShareLaTeX document and wrote whenever we could.

[4]http://botnik.org/apps/

## References

Boon, M. L. 2017. Augmenting Media Literacy with Automatic Characterization of News along Pragmatic Dimensions. In *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (pp. 49-52). ACM.

Ciarlini, A. E., Barbosa, S. D., Casanova, M. A., & Furtado, A. L. 2009. Event relations in plan-based plot composition. *Computers in Entertainment (CIE)*. 7(4), 55.

Doshi, L., Krauss, J., Nann, S., & Gloor, P. 2010. Predicting movie prices through dynamic social network analysis. *Procedia-Social and Behavioral Sciences*. 2(4), 6423-6433.

Oghina, A., Breuss, M., Tsagkias, M., & de Rijke, M. 2012. Predicting imdb movie ratings using social media. In *European Conference on Information Retrieval* (pp. 503-507). Springer, Berlin, Heidelberg.

Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. 2012. A literature review and classification of recommender systems research. *Expert Systems with Applications*. 39(11), 10059-10072.

Reagan, A. J., Mitchell, L., Kiley, D., Danforth, C. M., & Dodds, P. S. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*. 5(1), 31.