Unboxd: a Movie Recommendation System

Christine Chen[1], Dhruv Chavan[2], Eric Li[3], Eric Zhang[4]

Computer Science Department
Luddy School of Informatics, Computing, and Engineering
Indiana University Bloomington

[1]`cch8@iu.edu`
[2]`dvchavan@iu.edu`
[3]`ermili@iu.edu`
[4]`ez2@iu.edu`

Abstract— Recommendation systems are widely used to personalize and enhance user experiences, and they are especially useful on media review platforms like Letterboxd and IMDb. This project mimics these platforms and develops a recommendation system that suggests trending movies that align with a user's previously watched movies and preferences. It utilizes an international movies dataset from Kaggle, author Eric Li's Letterboxd reviews and ratings, and a trending movies dataset from Kaggle. The recommendation system has two components. First, the recommendation system obtains a movie most representative of a user's history and taste based on the movies they have rated and reviewed on Letterboxd. Second, the system uses this representative movie to suggest similar movies that were trending at one point. These two components utilize data mining techniques including sentiment analysis, cosine similarity, and k-nearest neighbors. The system aims to produce accurate movie recommendations for movie watchers of various experiences.

Keywords— Recommendation System, Letterboxd, Sentiment Analysis, Cosine Similarity, K-Nearest Neighbors

I. Introduction

Recommendation systems are essential to digital platforms, enabling personalized content and improving user experience. Movie recommendations help users discover new films that align with their preferences by analyzing past behaviors and movie characteristics. This project aims to design a system using data mining techniques that generates trending movie recommendations similar to the movies a user has previously both rated and reviewed, with a bias towards movies the user liked.

A. Related Works

Data from movie ratings platforms, such as IMDB, have been used to predict box-office outcomes [2, 5]. Although these methods [2, 5] could be used to recommend upcoming movies predicted to have successful box-office outcomes, the recommendations would be the same for each individual using them. Hence, there are several traditional approaches for personalized recommendation systems [6]: content-based filtering, collaborative filtering, and hybrid filtering. Content-based filtering systems recommend based on a user's history. Collaborative filtering systems recommend based on the content liked by users with similar profiles. Hybrid filtering systems combine content-based filtering with collaborative filtering. This paper pursues content-based filtering over the other methods due to limitations with gathering user data.

Content-based filtering suffers from the cold-start problem [3]. New users of a system have little history to draw from to curate accurate recommendations. To further increase recommendation accuracy, and decrease the computation required to produce recommendations, movie information feature reduction has been suggested and pursued [1, 3, 4]. Particular methods include single-value decomposition (SVD) [1, 6], and principal component analysis (PCA) [3]. Additionally, sentiment analysis, using a valence aware dictionary and sentiment reasoner (VADER) model, has been employed [4] on Twitter posts about movies. Thereafter, the results have been used in a hybrid filtering system as numerical weights for determining recommendations.

II. Methods

A. Data Gathering

For our purposes, we required two datasets: a dataset with information about the movies watched by an individual, including personal ratings and reviews, and another dataset of trending movies. In order to ensure meaningful comparisons between instances of the datasets, the two datasets needed to have matching features. Thus, the following features were extracted from a movies dataset ("International")[1] from Kaggle: *original_title*, *release_year*, *imdb_id*, *runtime_seconds*, *certificate_rating*, *genres*, *plot*, *directors*, *writers*, *actors*, *companies*, and *poster_url*. The dataset considered 950,000 films spanning from 1874-2025.

The first dataset is initialized on a per user basis. It is done by scraping Letterboxd, using the Python library of the browser automation tool *Selenium*[2], to get a user's reviews and ratings for movies. The second dataset was initialized using a trending movies dataset ("Trending")[3] from Kaggle. For both datasets, the movie titles and release years were used to retrieve features pertaining to those movies from the International dataset.

TABLE 1

INSTANCE FROM THE INTERNATIONAL DATASET WITH A SUBSET OF ITS FEATURES

original_title	**release_year**	**certificate_rating**	**genres**	**plot**	**directors**	**actors**
Longlegs	2024	R	Crime, Horror, Mystery, Thriller	In pursuit of a serial killer, an FBI agent uncovers a series of occult clues that she must solve to end his terrifying killing spree.	Osgood Perkins	Maika Monroe, Nicolas Cage, Blair Underwood, Alicia Witt

To showcase the recommendation system, we exhibit the results, intermediate and final, of author Eric Li's usage of the recommendation system.

TABLE 2

Subset of author Eric Li's scraped Letterboxd data

original_title	**release_year**	**user_rating**	**user_review**
Five Nights at Freddy's 2	2025	3.5	Chica where are you I need you
Don't Move	2024	1.0	Don't watch 🙏
Longlegs	2024	3.5	Crimson or clover. Intensity In the scene where Maika Monroe sees Nicolas Cage as Longlegs for the first time, her resting heart rate of 76 BPM reached 170. Monroe's heartbeat plays over the interaction, underlining the intensity. Not just Monroe's heartbeat, but also mine while viewing the film, signifies the intensity of

[1] https://www.kaggle.com/datasets/pavan4kalyan/imdb-dataset-of-600k-international-movies

[2] https://www.selenium.dev/

[3] https://www.kaggle.com/datasets/amitksingh2103/trending-movies-over-the-years

			Longlegs. The camera work, specifically the dolly zooms elevated the uncertainty of what was going to unfold. 90's Nostalgia From the start, director Oz Perkins immerses the viewer in a specific, time and place: the mid-1990s. The opening scene, in particular, leverages a desaturated nostalgia similar to classic serial killer thrillers. The plot is faithful to the accuracy of how things worked during that time, especially within the context of the mystery genre. While this gave the story an authentic feel, the ending felt a little underwhelming. Other If you are new to horror films, I would definitely not recommend watching Longlegs on your birthday. I didn't think I would be able to take Nicholas Cage seriously, but he definitely fit his role.
Midsommar	2019	2.5	It's fine. It's Sweden. Ambience The prolonged static shots and disorienting hazy visuals create the unsettling, dream-like aesthetic and convey the characters' drug-induced states. The vibrant flowers/background pop against the characters' white outfits. At one point, my sister was practicing Chopin's Valse to Madame the Countess Delphine Potocka downstairs, adding an unexpected ambiance and tricked my friends and me into thinking it was part of the soundtrack. Ari Aster If Ari Aster did not direct Midsommar, I'm not sure its reputation would…

B. Data Preprocessing

The International dataset was subjected to a multi-stage preprocessing pipeline to transform all heterogeneous features into a dense, numerical representation suitable for the k-NN recommendation algorithm. A primary focus of the implementation involved mitigating the exponential increase in dimensionality and associated memory consumption caused by encoding high-cardinality features. The key Python libraries used for this transformation were *pandas* for data manipulation, *numpy* for numerical operations, and *scikit-learn* for feature engineering and scaling through *TfidfVectorizer*, *MultiLabelBinarizer*, and *StandardScaler*.

The initial phase involved structural cleanup and memory optimization. Irrelevant metadata columns, such as *original_title*, *clean_title*, and *poster_url* were dropped from the DataFrame. The intrinsic numeric features, *release_year* and *runtime_seconds*, were explicitly downcast to the float32 data type to reduce the baseline memory footprint prior to feature creation.

List-valued features, specifically genres, directors, writers, actors, and companies, were converted into numerical features using *MultiLabelBinarizer*. This process created a binary column for every unique item in the lists. To manage the substantial dimensionality inherent in these features, a constrained approach was implemented to avoid the creation of ten thousand or more columns for each unique feature. Top-K filtering was used on each list column where only the K most frequently occurring labels were retained and encoded. This threshold was set aggressively low for high-cardinality features to drastically reduce the feature space.

The *certificate_rating* column was prepared by imputing missing values with the string ‘Unknown’ and was then processed using standard One-Hot Encoding through *pd.get_dummies* in *pandas*. Consistent with the efficiency strategy, the output DataFrame was generated with the *sparse=True* parameter to ensure the new indicator columns utilized the memory-efficient *SparseDtype*.

The plot feature that served as the movie summary, was transformed into a numerical vector using TF-IDF vectorization through the *TfidVectorizer*. The vectorizer was configured to remove standard English stop-words. Due to the vast amount of unique words in the English dictionary excluding stop-words, feature generation was strictly limited to a maximum of 75 terms, and a minimum document frequency of 50 was enforced to exclude very rare or non-discriminatory terms. The TF-IDF weight $w_{i,j}$ for term t_i in document d_j is calculated as the product of its term frequency ($tf_{i,j}$) and its inverse document frequency (idf_i). The resulting feature set was integrated into the main DataFrame as a sparse matrix to maintain memory efficiency.

Quantitative features in *release_year* and *runtime_seconds* were standardized. Any remaining *NaN* values were replaced with the mean of the respective column. The features were scaled using the *StandardScaler*. This procedure standardized the features to have a mean of 0 and a standard deviation of 1, ensuring that features with larger original magnitudes do not dominate the distance metrics used by the k-NN algorithm in the recommendation system. The resulting preprocessed dataset contains only numerical and sparse binary features, which were combined with the features in the Trending dataset to improve the accuracy of the k-NN algorithm.

C. Representative Selection

Sentiment analysis was performed on each user review using natural language processing. We initially experimented with the TextBlob python library. However, after some tweaking and adjustments, the results did not accurately reflect the tone and the sentiment of the reviews.

To overcome this limitation, we transitioned to VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analysis. After some testing and adjusting, VADER proved to be better handling movie reviews. Using VADER, it generates a compound sentiment score between -1 and 1, representing the overall sentiment of the review.

TABLE 3
AUTHOR ERIC LI’S SCRAPED LETTERBOXD DATA POST SENTIMENT ANALYSIS

original_title	**release_year**	**user_rating**	**compound**
Five Nights at Freddy’s 2	2025	3.5	0.0
Don’t Move	2024	1.0	0.0
Longlegs	2024	3.5	-0.2881
Midsommar	2019	2.5	0.7807

The set of n movies watched by the user was iterated through n^2 times. In this process, for each of the n movies, a weighted sum of the cosine similarity between the movie and every other movie in the set was computed using *sklearn.metrics.pairwise.cosine_similarity*. In each computation of cosine similarity between movie instances, the weight assigned to the ‘other’ movie under consideration was used. The weight for a movie was computed by adding the *user_rating*, scaled via *StandardScaler*, to the *compound*.

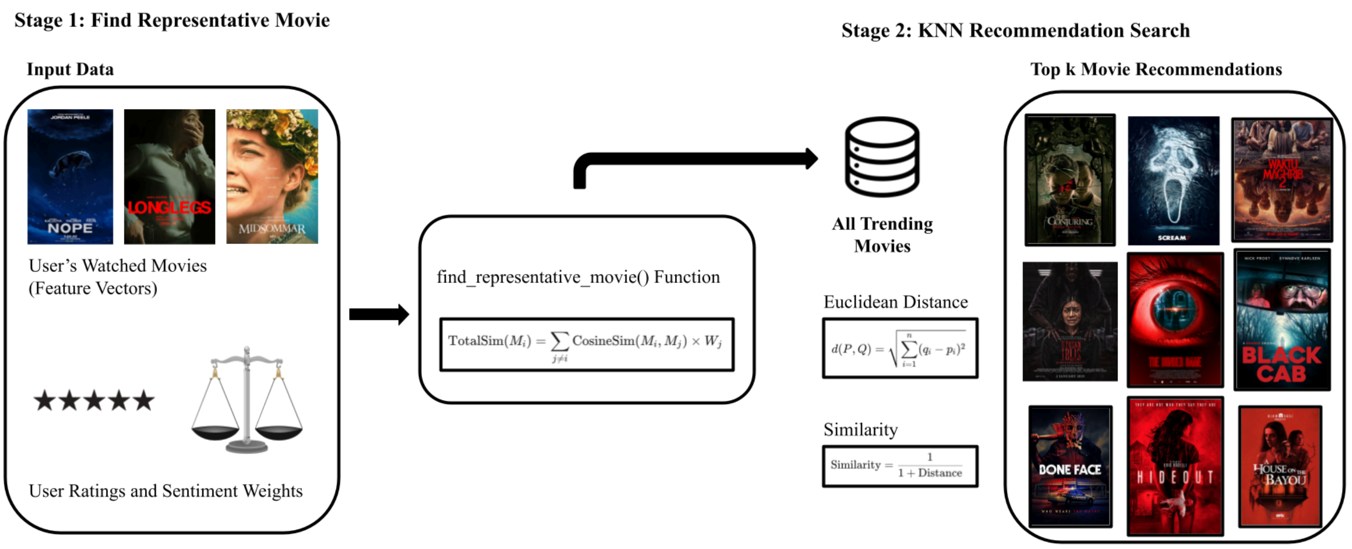
The representative movie was selected as the movie with the largest weighted sum of cosine similarities. This approach aimed to select a movie that aligned with a user’s preferences, and was also similar to other movies watched by the user.

TABLE 4
AUTHOR ERIC LI’S REPRESENTATIVE MOVIE WITH A SUBSET OF ITS FEATURES

original_title	release_year	certificate_rating	genres	plot	directors	actors
Midsommar	2019	R	Drama, Horror, Mystery, Thriller	A couple travels to Northern Europe to visit a rural hometown's fabled Swedish mid-summer festival. What begins as an idyllic retreat quickly devolves into an increasingly violent and bizarre competition at the hands of a pagan cult.	Ari Aster	Florence Pugh, Jack Reynor, Vilhelm Blomgren, William Jackson Harper

D. Trending Movie Suggestion

A k-NN algorithm using Euclidean distance was employed on the Trending dataset to find the *k* most similar movies to the representative movie. All *k* movies served as the trending movie recommendations for the user the representative movie was obtained for.



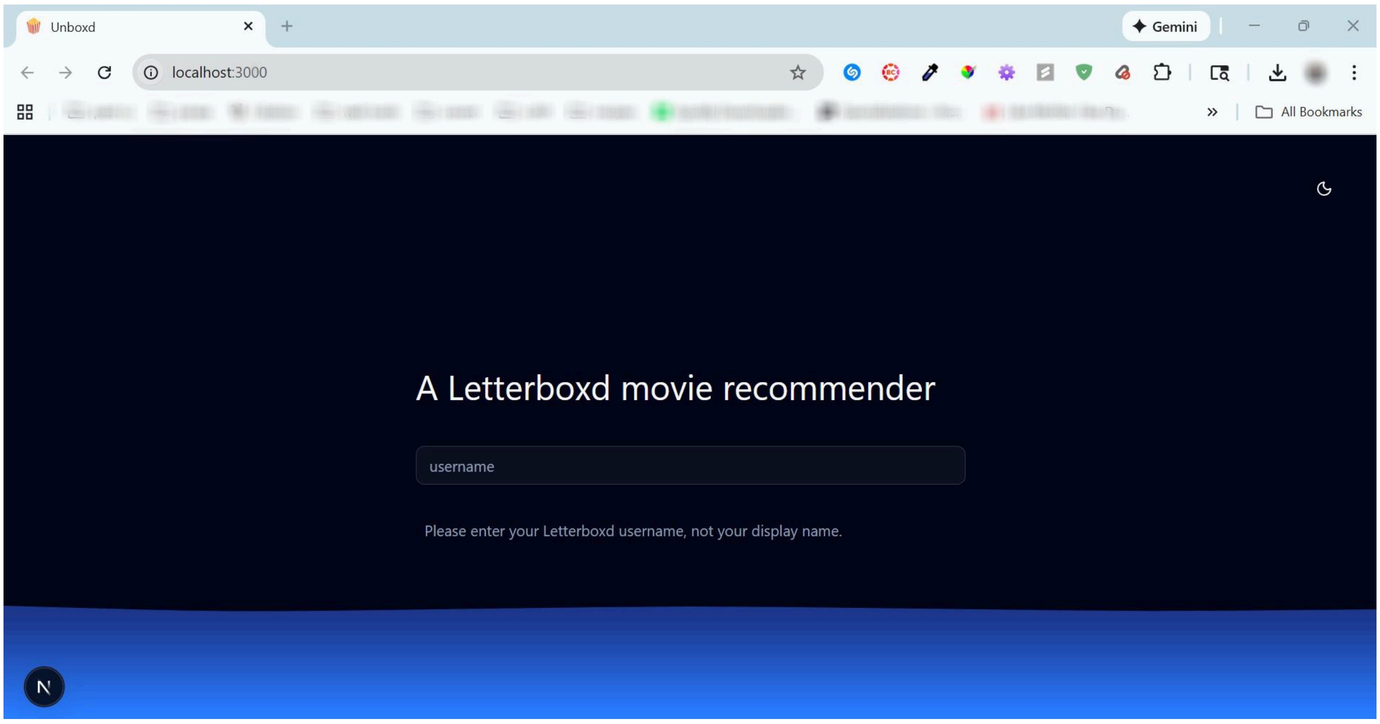
Stage 1: Find Representative Movie
Input Data
User's Watched Movies (Feature Vectors)
★★★★★
User Ratings and Sentiment Weights
find_representative_movie() Function
$\mathrm{TotalSim}(M_i) = \sum_{j \neq i} \mathrm{CosineSim}(M_i, M_j) \times W_j$
Stage 2: KNN Recommendation Search
All Trending Movies
Euclidean Distance
$d(P,Q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$
Similarity
$\mathrm{Similarity} = \frac{1}{1 + \mathrm{Distance}}$
Top k Movie Recommendations

Fig. 1 Diagram of recommendation system

E. Frontend UI

The frontend was developed using Next.js[4] and Shadcn.ui[5] components. FastAPI[6] was used to connect the frontend and the backend, and HeyAPI[7] was used to map the data models from the backend to the frontend.

[4] https://nextjs.org/
[5] https://ui.shadcn.com/
[6] https://fastapi.tiangolo.com/
[7] https://heyapi.dev/


Unboxd

localhost:3000

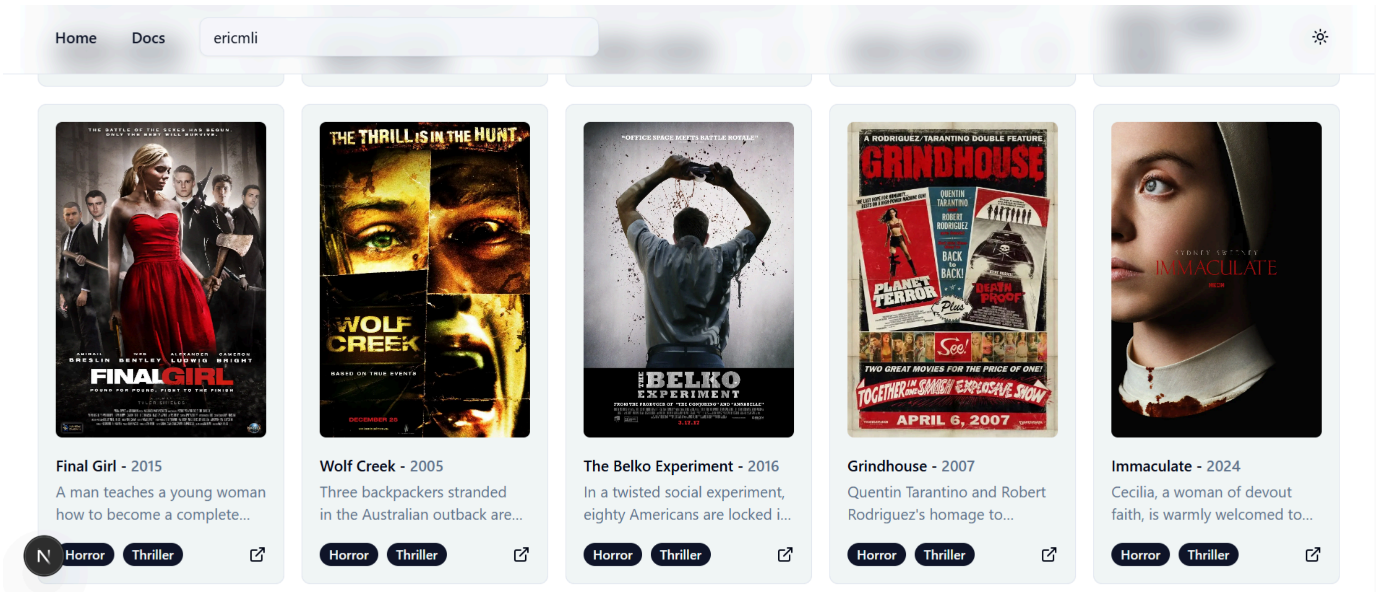
All Bookmarks

A Letterboxd movie recommender

username

Please enter your Letterboxd username, not your display name.

Fig. 2 Front page


Home Docs ericmli

Final Girl - 2015
A man teaches a young woman how to become a complete...
Horror Thriller

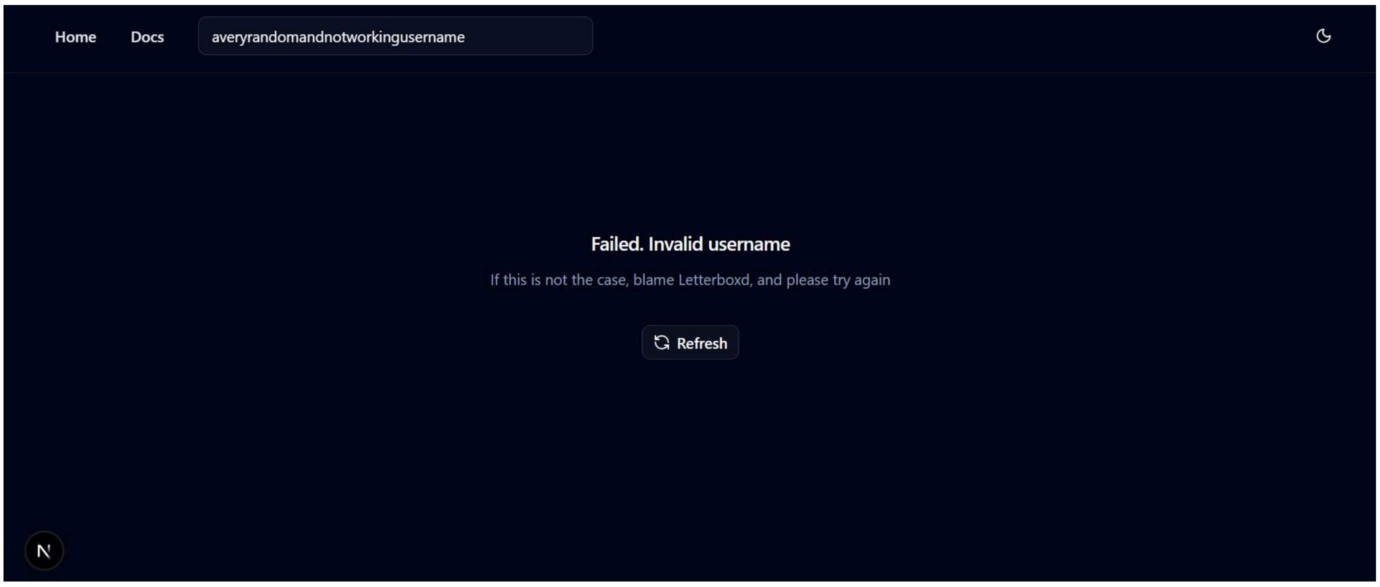
Wolf Creek - 2005
Three backpackers stranded in the Australian outback are...
Horror Thriller

The Belko Experiment - 2016
In a twisted social experiment, eighty Americans are locked i...
Horror Thriller

Grindhouse - 2007
Quentin Tarantino and Robert Rodriguez's homage to...
Horror Thriller

Immaculate - 2024
Cecilia, a woman of devout faith, is warmly welcomed to...
Horror Thriller

Fig. 3 Recommendations page, light mode


Home
Docs
averyrandomandnotworkingusername

Failed. Invalid username

If this is not the case, blame Letterboxd, and please try again

Fig. 4 Error page

III. Results

TABLE 5
Subset of author Eric Li's trending movie recommendations

original_title	**release_year**	**genres**	**plot**	**similarity_ score**
The Conjuring: Last Rites	2025	Horror, Mystery, Thriller	Paranormal investigators Ed and Lorraine Warren take on one last terrifying case involving mysterious entities they must confront.	0.3660
Black Cab	2024	Horror, Thriller	A couple's jovial cab driver diverts them to a remote, haunted road, revealing disturbing motives and his true intentions.	0.3658
Waktu Maghrib 2	2025	Horror, Mystery, Thriller	The rise and terror of the evil genie, namely Jin Ummu Sibyan. The mystical events in this take place after a period of 20 years.	0.3660

To test the quality of the movie recommendations, author Eric Li watched *The Conjuring: Last Rites*, the movie with the highest similarity score with the chosen representative movie: *Midsommar*. While *Midsommar* had a 2.5/5 rating, the movie contained qualities, such as the mystery genre, director Ari Aster, as well as actors Florence Pugh and Will Poulter, that author Eric Li preferred. Similar to the representative and top recommended movie from our algorithm, the other *k-1* movies boasted similarity in terms of the horror and thriller genre. To his testimony, the k-NN search algorithm effectively picked his preferred movie genre: thriller. However, horror is not among his priority preferences. Author Eric Li rated *The Conjuring: Last Rites* 1.5/5 stars, signifying his disapproval for the movie's quality. While negative emotions arose from his viewing experience, author Eric Li indicated a liking towards *The Conjuring* series as a whole in contrast with this finale. While the recommendation system suggested a movie that produced a negative user rating, it succeeded in offering a movie within a series that produced positive user ratings in the past.

IV. Discussion

As shown by author Eric Li's trending movie recommendations in Table 4, the system may only recommend movies from one genre. For example, if the system determines that the user's overall taste is horror, the k-NN algorithm would only pick movies that are similar to that genre. This disregards if the user likes multiple genres, e.g. both horror and romance.

Additionally, all of the similarity scores between author Eric Li's trending movie recommendations and his representative movie hovered around 0.36.

A. Limitations

The recommendation system we explored in this paper suffers from multiple limitations due to the limitation of gathering user data.

One such limitation is that new users on Letterboxd or users without many rated and reviewed movies, may have inaccurate recommendations due to not having a rich reviews history to base recommendations upon.

Another limitation is that we only consider recommending movies. However, on Letterboxd, shows are considered in the rankings and can serve significance to one's preferences as to what they desire to watch. The datasets selected for use only considers movies, thus eliminating some meaningful data that could be used to train our model and constricting which instances to consider from one's watch history.

The recommendation system also falls somewhat short in the case that a user has extremely random taste, meaning all the movies they have watched and reviewed are equally suitable candidates to be the representative of the user's history and taste. In this case, it is possible that the trending movies most similar to the representative movie would not be suitable recommendations for the user. However, the user's random taste may be suggestive of that fact that they would enjoy any movie recommendation.

Another pitfall of the system is that it does not consider how similar two genres may be. The downside of using cosine similarity is that it looks for overlaps of 1's and ignores semantic meaning. For example, applying cosine similarity on genre vectors means that "Action" and "Thriller" will be counted as "dissimilar" as they are not 1:1 matches, but in reality "Action" and "Thriller" films may be similar.

Due to limitations with time, scraping all user data from Letterboxd to test our recommendation system was unfeasible, so we only tested with a single Letterboxd user's data: author Eric Li. Because of this, our paper's analysis of our results may not be the most accurate, and there is also the problem of this problem space being unsupervised learning, where there is no true "right answer" for the recommendation we give.

V. Conclusions

This recommendation system demonstrates how classification methods can be utilized to make recommendations in an industry setting. More work can be done to improve the accuracy and efficiency of the system.

A. Future Work

1) *Improving Speed of Backend*:

Further work can be done to improve the speed of the backend. This can be done in a number of different ways.

The euclidean distances and cosine similarities between all the preprocessed movies can be precomputed and cached. This would save the repeated calculations made for each time we get a user's recommendations. However, the trade-off is that we will need a large amount of memory to store these results.

The system can be changed to only account for ratings instead of both ratings and reviews. This way, data can be scraped without interacting with JavaScript-rendered content (the movie reviews). However, the downside is that we no longer consider reviews when it comes to weighting the movie preferences.

A user's recommendations can be cached up to a certain amount of time. This way, every time the user refreshes the page, the backend doesn't run over and over again if no data has been changed. However, in the case that the user decides to rate a large amount of movies in a short period of time, these changes may not be reflected in the movies that are recommended.

We can potentially speed up the merging of data through databases like SQLite.

2) *Frontend Improvements*:

Quality of life features, like a button to clear input results on the search bar so that users don't have to manually do it themselves, can be added to improve the overall experience of the web application. We can also add an export option so that the users can import the recommended movies into their Letterboxd account.

The frontend can also be improved to only load the page after all elements have been loaded in to prevent confusion for the users if there are elements missing.

3) *Deployment:*

The final phase of the project is actually deploying a release so that people outside of the project can actually test and use this application.

VI. Author Contributions

Eric Z and Eric L were responsible for the preparation and preprocessing of the datasets. Christine and Dhruv handled feature extraction. Christine developed the frontend UI. Dhruv handled the integration between the frontend and backend as well as the research. Christine, Dhruv, Eric L, and Eric Z contributed to paper writing.

References

[1] Bhowmick, H., Chatterjee, A., & Jaydip, S. (2021). Comprehensive Movie Recommendation System. arXiv preprint arXiv:2112.12463

[2] Hsu, PY., Shen, YH., Xie, XA. (2014). Predicting Movies User Ratings with Imdb Attributes. In: Miao, D., Pedrycz, W., Ślęzak, D., Peters, G., Hu, Q., Wang, R. (eds) Rough Sets and Knowledge Technology. RSKT 2014. Lecture Notes in Computer Science(), vol 8818. Springer, Cham. https://doi.org/10.1007/978-3-319-11740-9_41

[3] Jayalakshmi, S., Ganesh, N., Čep, R., & Senthil Murugan, J. (2022). Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. Sensors, 22(13), 4904. https://doi.org/10.3390/s22134904

[4] Kumar, S., Halder, SS., De, K., Roy, PP. (2018). Movie Recommendation System using Sentiment Analysis from Microblogging Data. arXiv preprint arXiv:1811.10804

[5] Lee, K., Park, J., Kim, I. *et al.* Predicting movie success with machine learning techniques: ways to improve accuracy. *Inf Syst Front* **20**, 577–588 (2018). https://doi.org/10.1007/s10796-016-9689-z

[6] R. H. Singh, S. Maurya, T. Tripathi, T. Narula, & G. Srivastav, "Movie recommendation system using cosine similarity and KNN," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 5, pp. 556–559, Jun. 2020. doi:10.35940/ijeat.e9666.069520