**CS4V98 Final Report**

**Introduction:**

Student Name: Janam Sujit Parikh (JSP180001)

Faculty Advisor: Dr Anurag Nagar

Recommender systems are the systems that are designed to recommend things to the user based on many different factors. These systems predict the products that the users are most likely to purchase. Companies like Netflix, Amazon, etc. use recommender systems to help their users to identify the correct product or movies for them.

Recommender systems handles the volume of information present by filtering it based on the user’s preference and item similarity. It finds out the predicted rating for an item for a user.

There are two main types of recommender systems:

* Collaborative Filtering

The collaborative filtering method is based on gathering and analyzing data on user’s behavior. This includes the user’s online activities and predicting what they will like based on the similarity with other users.

* Content-Based Filtering

Content-based filtering methods are based on the description of a product and a profile of the user’s preferred choices. In this recommendation system, products are described using keywords, and a user profile is built to express the kind of item this user likes.

The goal of this project is to build a hybrid recommender system using feature hierarchies. Currently, the focus is on building a recommender system for movies. A fixed number of NLP concepts will be retrieved from the movie synopses. Then, a matrix of movie vs concepts will be created, and similarity between movies will be calculated using cosine similarity. This similarity will account for the content-based filtering aspect of the recommender system. We will also use user ratings in the algorithm to perform collaborative filtering.

**Progress Report:**

Github Repository: <https://github.com/jpspecter/MovieRecommenderSystem>

Datasets Used:

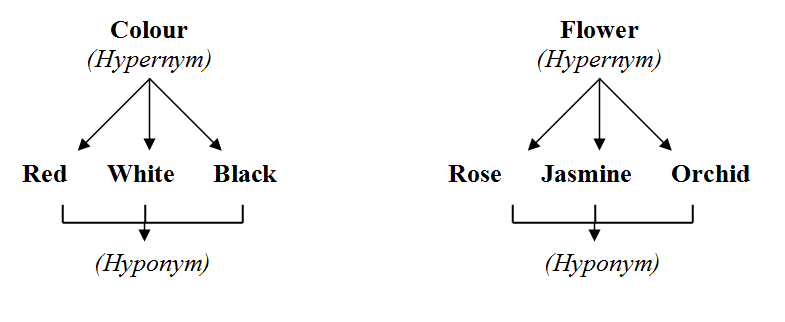
* CMU Movie Summary Corpus: <http://www.ark.cs.cmu.edu/personas/>
* MovieLens: <https://grouplens.org/datasets/movielens/>
* Rotten Tomatoes: <https://www.kaggle.com/datasets/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset>

First, the task was to understand and explore recommender systems and the MovieLens dataset. To do so, a basic recommender system was created that uses movie genres to perform content-based filtering. TF-IDF was also leveraged to capture the important genres of each movie by giving a higher weight to the less frequent genres. This recommender system returned the 10 most similar movies to the input:

A screenshot of a computer

Description automatically generated with medium confidence

Now that familiarity with recommender systems had been gained, the next task was to determine the NLP feature(s) that will be extracted to perform content-based filtering. A hypernym is a semantic field that is used to denote the superordinate of a specific word. For example, the hypernym of red, white, and black is color.



Since the goal of feature extraction was to return generalized concepts, hypernyms would be the ideal feature to extract. Next, an algorithm was hypothesized that would combine concepts from content-based and collaborative filtering. This will be used to predict the rating of a movie for a particular user. It will be based on the weighted average of movies already rated by the user by multiplying it with the similarity between target movie and already rated movies.

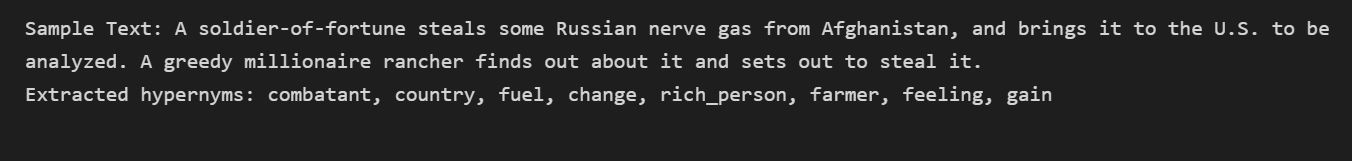
For example, rating (userA, movie10)

= ∑ rating(userA, moviei) \* sim(moviei, movie10) where movie is a moviei that userA has rated.

The goal of this project is to determine whether this approach of combining content-based filtering with collaborative filtering works by implementing a recommender system for movies that utilizes the algorithm.

* **Approach 1: Using movie summaries**

First, a python script was created to extract hypernyms from movie summaries in the CMU Movie Summary Corpus. These are the concepts that will be used to perform content-based filtering. The NLTK package was used along with WordNet to extract NLP features. For example,



Then, 100 most frequent hypernyms were identified from all the movie synopses. These hypernyms were used to create the movie vs hypernym matrix. The values in the matrix are either 1 or 0 depending on whether the synopsis of the movie contains that hypernym or not. Then, similarity between movies is calculated using cosine similarity. This concludes the content-based filtering part of the recommender system.

Graphical user interface

Description automatically generated

Unfortunately, the corresponding ratings dataset was not available. As you can see in the algorithm, ratings values for movies that the user has already rated in the past are needed. There was no dataset available that had movie synopses along with user ratings. So, after discussing this issue with my faculty advisor, we decided to take a different approach.

However, at the later stages of the project, we decide to try this approach again. Movie titles were matched from the CMU Movie Summary Corpus and the MovieLens dataset (https://grouplens.org/datasets/movielens/). The former dataset contained movie summaries and the latter contained user ratings. After synthetically creating a dataset containing movie summaries and user ratings, a complete recommender system that predicted user ratings was created. However, the accuracy of the system was very low (19.78%). This was because the hypernyms being generated from movie summaries were very general. We needed to extract deeper NLP features to determine the correct movie-movie similarity.

* **Approach 2: Using user ratings**

First, we determined a good dataset for our use-case. We decided to use the rotten tomatoes dataset as it contains textual user reviews along with numerical user ratings. Hypernyms were extracted from user reviews (instead of movie summaries, as in the previous approach). As before, Cosine similarity between movies was also calculated. To perform collaborative filtering, a user-rating matrix was created. Thus, we were able to create an end-to-end implementation of the recommender system and could put it to test.

Unfortunately, the predictions generated by the recommender system were not accurate. The accuracy of the system was 15.24%. This is because user reviews contain descriptive words about the experience of watching the movie. They might not contain words that describe the contents of the movie. For example, users may have described two different movies with similar words like ‘interesting’ or ‘captivating’. Thus, we got very high similarity value for dissimilar movies as users have used similar words to describe their experience of watching the movie. For example, ‘A Beautiful Mind’ is a historical drama whereas ‘The Expendables’ is an action movie. Both the movies vastly differ from one another in their content, tone, and style. However, their similarity value comes out to be very high.

Graphical user interface, application

Description automatically generated

Thus, extracting concepts from user reviews did not work for content-based filtering. We decided to take a different approach for performing the same.

* **Approach 3: Using user-generated tags**

Instead of generating concepts for doing content-based filtering, a decision was taken to use user-generated tags for the same. The dataset used was the MovieLens dataset. Some regular expression was also performed on the tags to make it uniform. As before, movie similarity was calculated, and a user-rating matrix was created. Again, an end-to-end implementation of our system was created, and it could be put to test. The system was not able to predict accurate ratings as there were many unique tags. The accuracy of the system was low (21.23%). Since the tags are user generated, there are many tags that describe similar concepts but are spelt or written differently. For example, if users wanted to indicate that the movie was of science fiction genre, they could tag it as ‘sci fi’, ‘sci-fi’, ‘science fiction’ etc. Thus, movies with similar plots had a low cosine similarity value due to the low number of common tags. A large majority of the movie pairs returned 0 as the similarity value. Thus, user generated tags were also not suitable for content-based filtering.

Graphical user interface

Description automatically generated with medium confidence

* Approach 4: Using keywords:

Next, keywords were automatically extracted from movie summaries as features for determining movie-movie similarity. A feature was used within the NLTK package that allows users to extract the key words from a text, and it even assigns scores to each word. This package is called Rake. After extracting keywords, they were vectorized using the CountVectorizer. Once we had the matrix containing the count for each word, the cosine similarity function was applied. This concluded the content-based filtering aspect. To perform collaborative filtering, the CMU Movie Summary Corpus and the MovieLens dataset (containing user ratings) were matched based on movie title. After testing the system, it gave us significantly better accuracy results (30.39%). This was still far from desirable, but it was step in the correct direction.

* Approach 5: Using keywords and TF-IDF:

Since the previous approach resulted in a better performance, keywords were again extracted from movie summaries. Rake also assigns a score to each keyword. Using that score, vectorization was performed with TF-IDF weights using the tfidfCountVectorizer. Upon testing this system, we got a better accuracy value than before (35.74%).

**Results:**

Following are the results obtained using the various approaches:

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Dataset | Accuracy | Possible Issues |
| Extracting WordNet hypernyms from movie summaries | CMU Movie Summary Corpus + MovieLens | 19.78% | Extracted hypernyms very generalized |
| Extracting WordNet hypernyms from user reviews | Rotten Tomatoes | 15.24% | User reviews contained similar descriptive words for different movies, high movie-movie similarity |
| Using user-generated tags | MovieLens | 21.23% | Large number of unique user-generated tags, low movie-movie similarity |
| Extracting keywords from movie summaries | CMU Movie Summary Corpus + MovieLens | 30.39% | Need to vectorize with TF-IDF scores |
| Extracting keywords from movie summaries and using TF-IDF | CMU Movie Summary Corpus + MovieLens | 35.74% | Need to extract deeper NLP features, improve prediction algorithm |

**Possible Solutions:**

The various approaches used to implement the recommender system have given subpar results. However, the efforts taken by us have given a specific direction for future work to be done on this project. The accuracy values significantly increased with each approach, indicating that a step in the right direction is being taken. Since we are taking a completely new approach of using a combination of content-based and collaborative filtering, it will take some time and effort to perfect an algorithm for the same.

Following are some possible solutions for implementing the algorithm:

* Extracting deeper NLP features like Named Entities, POS Tags, Syntactic parsing, Semantic parsing etc. from movie summaries. This will enable us in performing content-based filtering in an effective manner.
* Formulating a more comprehensive algorithm for predicting user ratings.