**Movie Recommender**

Justin Gilmer and Alan Benitez

CAP4601 Intro to Artificial Intelligence

**Abstract.**

In the following paragraphs, we will discuss the use of content-based filtration, the advantages and disadvantages of the filtration that is used, the methods that are implemented, the reasons that the methods were implemented, and how they successfully worked with our project. We will also discuss the structure of our project.

1. **Introduction**

We choose to do a movie recommender for our project which is basically a filtration program with the goal of predicting what other movies a user may like. There are many types of filtering that can be used to implement a recommender system and we decided that we would use a content-based filter. In our project, we use the TMDB 5000 Movie dataset from TheMoviedb.org[1] as the resource.

* 1. **Creating the DataFrame**

The first part of our project is the creation of the dataframe that is used throughout it. We started off by creating our Credits and Movies data frame by reading in the (tmdb\_5000\_credits.csv) and (tmdb\_5000\_movies.csv). We then merge our two data frames to create a new dataframe.

* 1. **Calculating the Score Column**

The next part of our project is the calculation and creation of the score column. We did this by computing the mean of the vote\_average column and taking the quantile of the vote\_count column which is used to calculate the weighted ratings for the movies. We then copied the dataframe while having the vote\_count column greater than or equal to the quantile version of the vote\_count. Next, we use the weighted ratings to create a new column called score for each row. Lastly, sorted each row by the new column in descending order.

* 1. **Creating the Top 10 Movie Graph**

Next is the creation of our top ten movie graph. This is done by defining the plot method. We defined the plot method by using the mathpltlib library and defining a variable to sort the data frame by the popularity column in descending order. We then changed the figure size and then defined the x and y axis. We defined the x axis as the top ten movie titles using the column “title\_y” and the y axis as popularity using the popularity column. We used the head(10) function for both columns in order to populate the graph. After the graph was populated, we gave the graph a title, labeled the x axis, and inverted the y axis.

* 1. **Implementing the content-based filter**

Finally, we implemented the content-based filter in order to recommend movies to users. We did this by first creating a matrix that removes all the stop-words such as “the”, “a”, “an”, and “in” with stop\_words. Afterwards, we replaced all the NULL values in the “overview” column with “” and used the fit\_transform method to put the “overview” column in the matrix. Next, we defined cosine similarity using the linear kernel function on the matrix created from the “overview” column. Following that we created the indices by using the titles of the movies. Lastly, we defined the get\_recommendations function by enumerating the cosine similarities to tuples and sorting them in descending order.

1. **Methods / Theory**

We had to study content-based filtering and cosine similarity. Content-based filtering uses cosine similarity to recommend movies that have similar lead actors, directors, and genre. The cosine similarity that is used for content-based filtering is based on keyword overlaps. TF-IDF is used to compute the overall importance of the keywords by measuring the frequency and weights of them. TF-IDF computes the importance of the keywords using

*TF-IDF(i,j) = TF(i,j) \* IDF(i) [2]*

where TF is term frequency, IDF is inverse document frequency, i is the keyword, and j is the document. TF(i,j) is calculated using

*TF(i,j) = [2]*

Freq(i,j) is the number of occurrences of the keywords in the document and maxOthers(i,j) denotes the highest number of occurrences of another keyword. IDF(i) is calculated using

*IDF(i) = [2]*

N is the number of all recommendable documents and n(i) is the number of documents in which keyword i appears.

1. **Discussions**

Content-based filtering can be used for recommendations, but there are some slight disadvantages to it. Some of the disadvantages to content-based filter are that the user does not get exposure to other types of movies and that it can not be expanded unless the user likes different types of movies.[3] Another issue with content-based filtering is that semantic meanings remain unknown to the program.[2] For example; it cannot tell if the usage of a word is used in a negative context. The strengths of content-based filtering are that if recommendations are wanted for a specific category such as genre of films or lead actors this is easily determined.

1. **Conclusions**

In conclusion, content-based filtering can be used for creating a simple recommendation system, but there are other types of filtrations that can be used instead to create a more concise recommendation system. Content-based filtering cannot determine the semantics that the keyword is used in and does not give the exposure that other filtrations can give users. Therefore, it could be better to use content-based filtering in pair with another type of filtration such as collaborative filtering.

**References**

1. TMDB 5000 Movie dataset download, <https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?resource=download>, last accessed 2022/04/23
2. Tutorial: Recommender Systems, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.592.1030&rep=rep1&type=pdf>, last accessed 2022/04/24
3. Create Your Own Movie Recommendation System, [Build A Movie Recommendation System on Your Own (analyticsvidhya.com)](https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/), last accessed 2022/04/23