Movie Recommendation System

Introduction to Artificial Intelligence: Final Project

Jennifer Margaret Manful- MIS'24

Abigail Efua Tetteh - CS'24

Ashesi University

CS524 Introduction to Artificial Intelligence

Cohort A

Group 12

Ayokor Korsah, Ph.D.

December 12, 2022

Introduction

A recommendation system is an artificial intelligence or AI algorithm, usually associated with machine learning, that uses Big Data to Predict what users or customers want based on historical data. We have watched movies and gotten to the point where we have run out of ideas and struggled to find movies as exciting as those we have observed, which in most cases, we never get a replica of our favorite movies on various streams. As time-consuming and stressful as it sounds, we decided to work on the movie recommendation system to help users locate movies they may have never found.

Background

In creating an AI recommendations system, there are two main approaches to take, the first being collaborative filtering, which uses similarities between users and items simultaneously to provide recommendations. This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B. The second method, content-based filtering, uses item features to recommend other items similar to what the user likes based on their previous actions or explicit feedback.

For the problem of building a movie recommendation system, collaborative filtering is best suited; and under collaborative filtering using Item to item filtering, which matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list.

For our movie recommendation, AI, a term frequency-inverse document frequency TF–IDF (term frequency-inverse document frequency), is a term weighting scheme commonly used to represent textual documents as vectors (for classification, clustering, visualization, retrieval, etc.).

Automatically locating keywords that best reflect the document's content is a crucial task in document analysis. The TF–IDF heuristic-based method is one of the most used methods for keyword discovery.

How can we find a document's keywords given only the document? In many situations, this is simple: for instance, if a document frequently discusses machine learning, the term "Machine learning" should be chosen as a keyword. Although "a" or "the" frequently occur in documents, we should use them as keywords characterizing a particular document since they do so frequently. Of course, this does not imply that any word frequently appearing in a text is a significant keyword. Therefore, it's important to consider the frequency of a word's occurrence in other papers and its frequency in a given document to find keywords.

A unique TF-IDF approach is used in information retrieval and text mining to automatically identify keywords (Rajaraman and Ullman 2011). The concepts underlying this algorithm were initially put forward by Jones (1972). This technique uses the following three numerical characteristics: the total number N of documents in a given corpus D; the number of(t) of documents that contain the given term t; and the term frequency, which is the frequency with which a given the word appears in a document (Tf(t,d)).

In information retrieval and related fields, cosine similarity is an extensively used statistic. This measurement represents a text file as a vector of phrases. According to this concept, the cosine value between the word vectors of two papers can be used to determine how similar they are [5]. Any two texts can be used to implement this metric (sentence, paragraph, or whole document). The degree of similarity between user queries and documents is ranked from highest to lowest in

search engine cases. A greater similarity score between the term vectors of the document and the query indicates more relevance between the document and the query.

Although it should consider the word's cosine similarity, cosine similarity for similarity measurement between document and user query still needs to be able to manage the semantic meaning of the text fully. Cosine similarity measurements between two-term vectors implemented syntactically can produce inaccurate results. The issue of semantic meaning may still need to be resolved by syntactic matching. to continue the procedure. 1.e., an information retrieval system may yield erroneous results and suffer performance degradation.

APPROACH

Working on a recommendation system presents different approaches that cover many criteria upon which the system should make exciting suggestions. Some of these criteria are likes, ratings, comments, and previous history in terms of interaction with the product. Building a user-to-user recommendation system is a fantastic recommendation system capable of enhancing its efficiency. New users are added every time, and various interests are included in the data to recommend movies to other potential users. Although this was the first idea of our work, we had to drop it because we would have to build user profiles and create efficient means of keeping user interactions with the system. It will also mean getting more people to sign into the system to help us generate our dataset before building the actual recommendation system. The user-to-user system will not only have exposed us to the idea of system suggestions, but it would have also allowed us to learn how to manage user data better, but we had to drop it due to time and inadequate knowledge about the approach. With this, we settled on the item-to-item movie

recommendation system. The item-to-item recommendation system is the best approach because we will use an accumulated dataset to help us suggest the best movies based on movie ratings, genres, and tags of other users who may have the same interest.

Finding and cleaning the dataset: we found the dataset on grouplens MovieLens |GroupLens.This dataset (ml-25m) describes a 5-star rating and free-text tagging activity from [MovieLens](http://movielens.org), a movie recommendation service. It contains 25000095 ratings and 1093360 tag applications across 62423 movies. These data were created by 162541 users between January 09, 1995, and November 21, 2019. This dataset was generated on November 21, 2019. From the ml-25m folder, we used movies.csv and ratings.csv. Next, we loaded the dataset onto our google colab. We cleaned the title column utilizing the regex library, which allowed us to remove any commas, dashes, and parentheses from the title. We also created a new column to keep the cleaned tags. We have also used the Term Frequency- Inverse Document Frequency (TF-IDF) to get special terms from the cleaned titles.

After obtaining and cleaning our data, we employed machine learning to build the system. We used a supervised learning approach because our dataset contained labeled data represented by columns. We only used two files from the folder from which we only used the movieId, userId, move title, and rating columns and dropped the remaining columns since they do not affect our model in any way. Under supervised learning, we used the K nearest neighbor's algorithm to find movies closest to the ones searched by a user. The k-nearest neighbor's algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier which uses proximity to make classifications or predictions about the grouping of an individual data point. The K in our system is determined by an angle found with the help of the cosine similarity library in python. The angle of the movie title represents the K, and other movies which are closer in angle are the ones

that are recommended. With the cosine similarity from sklearn, we created a function that allows us to compute the similarity between a term we enter. Cosine similarity is used with the same principle of cosine angles, where even if the similarity of content is less similar, it would be considered the least recommended content. The recommendations generated would be at the top for higher similarity of contents. Cosine similarity is also used in textual data to find the similarity between the vectorized texts from the original text document.

Ipywidgets and Ipython.display: With the cleaned data, we used the ipywidgets module in python to help us create a search box where users can type titles of movies they want the system to recommend. The ipywidgets are used along with the Ipython.display module to allow the system display the recommended movies to the user.

Recommendation functions: We have created tasks that will help us recommend movies based on the ratings that similar users and all users have liked. The recommendation function recommends movies that users like us have rated to give movies ideas available based on our interests. The function further creates a score for each movie recommended and displays it to the user based on the descending order of the scores. The recommendation focuses on the title and ratings and calculated scores based on the angles generated by the cosine similarity function to recommend movies.

RESULTS AND ANALYSIS.

Our code shows that the system suggests movies to users once they enter a movie title. The approach is also fast because the system displays outputs immediately after the user starts typing and consistently reorganizes them into the correct order once they are done typing their entry. To know how intelligent our algorithm is, we proceeded to type titles of locally produced, which

were not included in the data accumulation, to see if the system would give a result. Surprisingly, it worked as expected because it returned nothing since it has no information about the text, therefore can not suggest a movie to the user.

CONCLUSION

In conclusion, this report was aimed at recommending movies to users based on title and other users' ratings using item-to-item collaborative filtering. We built a system that produces movie suggestions for users based on other users' ratings who may have the same interest as us. Working on the project enabled us to understand data manipulation and expose ourselves to the limitless things we can do with the libraries we have used in class. Although we have achieved the goal of our project, we look forward to different ways of scaling it by including more criteria for the suggestion.

References

Dataset: https://grouplens.org/datasets/movielens/25m/

Havrlant, L., & Kreinovich, V. (2017). A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation). *International Journal of General Systems*, *46*(1), 27-36.

Rahutomo, F., Kitasuka, T., & Aritsugi, M. (2012, October). Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST* (Vol. 4, No. 1, p. 1).