Text

Description automatically generated

**Department of Computer Science**

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the Master of Science degree in Software Engineering.

|  |  |  |
| --- | --- | --- |
| **Movie Recommendation System** | | |
| Project Title | | |
| **Ayleen Chan** |
| Student Name |
| **Ning Chen, Ph.D.** |
| Advisor's Name |
|  | |
| Advisor's signature | Date |
|  |
| Reviewer's Name |
|  | |
| Reviewer's signature | Date |

**Abstract**

With the ever-increasing number of web applications available around the world, it's becoming more difficult to find the right information for a user within a short amount of time. Without a recommendation system, users would have had to spend a significant amount of time searching for the information they needed online. Recommendation systems are one of the most used applications of machine learning technologies in today's business. They are essential and an important part in e-commerce. Today, most of us heavily rely on online platforms, such as Amazon, Pinterest, Netflix, and Spotify, as their embedded recommendation systems can help us to find the product we want easily. Using the movie data from Kaggle.com, we created a movie recommendation system based on the similarity of the movies that users search. The system will recommend the movies based on the description of the input and additional features (casts or genres) that the user mentioned. There are main methods to approach the recommendation engines, namely collaborative and content-based filtering. Since we want to recommend using the additional information about users and/or items, the content-based method is the most appropriate and simple method to apply for this project.

The first section of the project will include the introduction, which will discuss the project's objective and the requirements of the development environment used to develop the project. The second section will include the functional and non-functional requirements of the project. The fourth section will contain the design description that demonstrates the data flow diagrams and a detailed explanation of the movie recommendation system. Lastly, the report will discuss the implementation of the overall application, along with a brief description of the development code file structure and will explain the project activities. In the Methods and Algorithm Discovery part, a detailed explanation of ***cosine similarity*** and ***content-based filtering method*** of the recommendation system will be discussed.

Contents

[1. Introduction 4](#_Toc103532311)

[1.1 Objective 5](#_Toc103532312)

[1.2 Development Environment 5](#_Toc103532313)

[2. Requirements Description 6](#_Toc103532314)

[2.1. Functional Requirements 6](#_Toc103532315)

[2.2. Non-Functional Requirements 6](#_Toc103532316)

[3. Choosing Dataset 7](#_Toc103532317)

[4. Methods and Algorithm Discovery 9](#_Toc103532318)

[4.1 Content-based Method 9](#_Toc103532319)

[4.2 Cosine Similarity 10](#_Toc103532320)

[5. Design Description 12](#_Toc103532321)

[5.1 Architecture and Design 12](#_Toc103532322)

[5.2 Internal Functions 12](#_Toc103532323)

[6. Implementation 13](#_Toc103532324)

[6.1. Phase I – Method and Algorithm Discovery Phase 13](#_Toc103532325)

[6.2. Phase II – Development 13](#_Toc103532326)

[6.2.1 Project Structure Description 14](#_Toc103532327)

[6.2.2 Development Results 16](#_Toc103532328)

[7. Operating Instructions 18](#_Toc103532329)

[8. Recommendations for Enhancements 20](#_Toc103532330)

[9. Conclusion 21](#_Toc103532331)

[10 . Acknowledgement 21](#_Toc103532332)

[11. GitHub Repository 21](#_Toc103532333)

[12. Bibliography 23](#_Toc103532334)

**Keywords:**

Content-based filtering**,** Collaborative filtering, Hybrid filtering**,** Cosine Similarity

# 1. Introduction

There are various techniques and implementations to build a recommendation system. Different developers use different approaches such as collaborative filtering, content-based filtering, and hybrid filtering (which combine both approaches) [4]. Collaborative filtering is a system that makes recommendations based on past interactions between the users and the system. Although the collaborative filtering method is mentioned as the most mature and commonly implemented technique, it has its own drawbacks. Since the collaborative filter method is based on past interaction with the user, it cannot make recommendations to new users or any new products to the users. A content-based method is an approach that makes recommendations based on the similarity of the products. The content-based method does not need the user's history or the user's search history for the product. Instead, Content-based requires the description of the item(product) since this approach uses the similarity of the products and additional information about the users or products. Compared to the collaborative filtering method, although the content-based approach suffers less from the cold start problem, the *content-based approach* requires more information about products' features. While the complexity of each of these approaches varies, complexity does not always equate to "excellent" performance. The simplest solutions and implementations often produce the most effective results [3].

In this project, the movie recommendation system will be built based on the similarity of the movies that user's input. The system will recommend the movies based on the description of the user's input and additional features that the user mentioned.

Although there are many recommendation systems, our primary purpose of developing a movie recommendation system is to better understand the recommendation systems and determine how effective they are using the machine learning methods described in the following section.

## 1.1 Objective

The main objective of this project is to develop a movie recommendation system using machine learning algorithms that can recommend similar movies based on the user inputs and based on the additional features of the movies. Beyond that, the goals of this development project are to understand and research similarity measurements in machine learning and provide an effective method for a movie recommendation system based on our users' requirements. Throughout the development and research process, the paper aims to explain why the content-based approach works best for our project and determine the right model for our movie recommendation system.

The final results of this project will include (1) a detailed explanation of machine learning algorithm selection, (2) a command-line application that can perform all the tasks described in this objectives section.

The final recommender system will be able to recommend similar movies to the user with a full range of functionality. For example, if the user search "Spider-Man" movie, the system will recommend other similar movies such as "the Avengers", "The Amazing Spider-Man", "Eternals", etc. The main goal of the final demonstration of the project is to show the program's full range of functionality efficiently.

## 1.2 Development Environment

The environment used to develop this project includes the following hardware and operating system:

1. Central Processing Unit: 2.0 GHz quad-core10th generation Intel Core i5

2. Random Access Memory: 16GB of 3733MHz LPDDR4X onboard memory

3. Graphical Card: Intel Iris Plus Graphics

4. Operating System: macOS Big Sur

The following software will be used in the development of the program:

1. Languages: Python, Jupyter Notebook
2. Editor: PyCharm
3. Dataset: Kaggle

# 2. Requirements Description

## 2.1. Functional Requirements

|  |  |
| --- | --- |
| No | Description |
| FR1 | The users shall be able to see similar movies based on what they search |
| FR2 | The user shall be able to see data visualization in Jupyter Notebook |
| FR3 | The user shall be able to search the similar movies based on movie title with additional features (cast and genres) |
| FR4 | The system shall display the top 10 similar movies based on the user search |
| FR5 | The system shall display available movies lists on Console |
| FR6 | The user shall be able to see data visualization in Jupyter Notebook |
| FR7 | The system shall provide similarity scores of the movies based on user search |
| FR8 | The system shall recommend movies based on content-based computations |
| FR9 | The system shall recommend movies based on movie title. |
| FR10 | The system shall recommend movies based on movie title with casts. |
| FR11 | The system shall recommend movies based on movie title with genres. |

## 2.2. Non-Functional Requirements

|  |  |
| --- | --- |
| No | Description |
| NFR1 | The system must have an accuracy of at least 80% |
| NFR2 | The system must be able to calculate functions within 3 seconds |
| NFR3 | The system must be able to display within 5 seconds |
| NFR4 | The user shall be able to access the recommendation system without failure |
| NFR5 | The system shall maintain system integrity |
| NFR6 | The system shall be user-friendly and intuitive |
| NFR7 | The system shall be portable to different operating systems |

# 3. Choosing Dataset

During the process of data training and choosing data for the movie recommendation system, Jupyter Notebook was used to analyze and visualize the data. The movie dataset used in this project was acquired from Kaggle.com, however the dataset was scraped from The Movie Database API [5]. After analyzing the datasets available on Kaggle.com, the tmdb\_5000\_credit dataset and the tmdb\_5000\_movies dataset were selected for the movie recommendation system. Although the movie dataset includes movie titles and genres, there is no column for casts. ***Figure 3.1*** is a snapshot of the visualization of the top 5 rows of the movie dataset that is displayed in jupyter notebook during the data analysis process. In this project, the user will be able to get a recommendation for similar movies based on the names, casts, and genres of the movies. Since the first dataset lacks information for casts, a second dataset (tmdb\_500\_credit) was utilized for the required data. The second dataset contains full credits for both casts, crews, and movie titles, but it lacks genre information. Figure 3.2 depicts a sample of the data contained inside the credit dataset.

|  |  |
| --- | --- |
| Table  Description automatically generated |  |
| Table  Description automatically generated with medium confidence |  |

**Figure 3.1 Movie Dataset**

|  |  |
| --- | --- |
|  | Text  Description automatically generated |

**Figure 3.2 Credit Dataset**

At last, the two datasets were combined to fill in the shortcomings of each dataset. The main advantage of using the movie dataset and credit dataset from Kaggle.com is that they have several columns that contain JSON. As both datasets lacked those columns, new columns (casts, directors, producers, and writers) were created using JSON columns. Since the movie recommendation system can recommend similar movies based on the additional features (casts and genres), the datasets that include JSON columns were beneficial for creating the required columns for this recommendation system.

# 4. Methods and Algorithm Discovery

## 4.1 Content-based Method

A content-based method is an approach that makes recommendations based on the similarity of the products. The content-based method does not need the user's history or the user's search history of the product. Instead, the Content-based requires the description of the item(product) since this approach uses the similarity of the products and additional information about the users or products. Compared to the collaborative filtering method, although the content-based approach suffers less from the cold start problem, the content-based approach requires more information about products' features.

The main idea of the content-based method is to create a model based on the available features. It can be based on users' features or item features. The recommendation can be made based on users' profiles. Based on the user's age and sex, the content-based model tends to make a better prediction. For example, a teen's preferences in movies will differ from an older person's. In this case scenario, the content-based method will be able to make a recommendation by collecting the information on the user profile [4].

In this project, the movie recommendation system will be built based on the similarity of the movies that the user search. The system will recommend the movies based on the description of the user's input and additional features that the user mentioned. Therefore, based on our requirements, the content-based method is the most appropriate and simple method to apply for this project.

## 4.2 Cosine Similarity

In order to recommend similar movies based on the user’s input, a content-based filtering method will be used, as mentioned in section 4.1. The idea of the content-based filtering method is to compare items to other items according to their features to calculate a similarity score. In that way, we will be able to provide a list of recommendations based on the user’s explicit input and its similarity score. In order to calculate the similarity score, cosine similarity will be used as the measure.

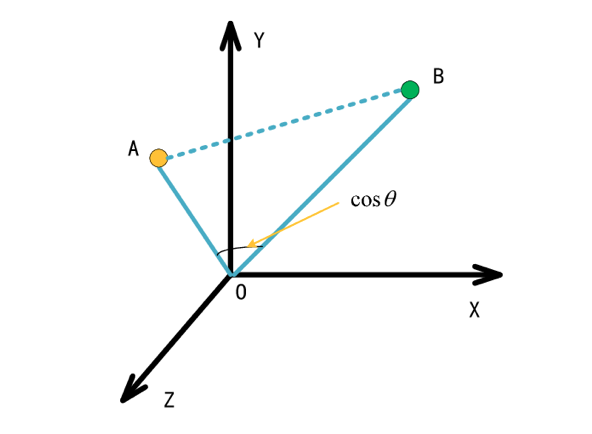
A picture containing diagram

Description automatically generated

where, n: number of items

Ai: the ith value of items vector A

Bi: the ith value of items vector B



***Figure 4.1 Similarity between vector A and B on 3-dimensional space. Ref: [6]***

The way cosine similarity works in this movie recommendation system is that the formula is dependent on the input of two items, vector A and vector B. When a user interacts with the recommender system by searching for a movie title in the dataset, the input text was then converted into a list of strings. The input can be simple text or text separated by commas. A list of the input string (movie title in our case) was used as an index to generate the movie overview so that the system could transform the movie overview as vectors. The movie overview was then preprocessed with NTLK preprocessing techniques. The preprocessing data (movie overview) is converted into a TF-IDF (Term Frequency Times Inverse Document Frequency) matrix, which is typically called as a vectorizing process in NLP. After transforming text into a matrix of counts, the cosine similarity formula will calculate the distance between two vectors The vector that is closest to the user's input vector is then considered to be the most similar item (movies). After applying cosine similarity to each movie in the dataset, the cosine similarity will give the result of filtering the similar movies based on the movies that the user searched. Finally, similar movies will be generated in the order of similarity scores.

# 5. Design Description

## 5.1 Architecture and Design

The data flow of the movie recommendation system is simple since project main concept is to develop a simple movie recommendation system that would recommend the top ten similar movies .The architecture of the data flow diagram can be seen in

***Figure 5.1.***

Diagram

Description automatically generated

***Figure 5.1 Data Flow of Content-based Movie Recommendation System***

## 5.2 Internal Functions

The project contains four main files to get a recommendation based on the user's requested movie.

* The first one is ***main.py***, which will act as the user interface when the program is started.
* The second file is ***recommender.py***, which will interact with other three files of the system.
* The third file is ***movie\_data\_report.py***, which will mainly perform the processing of all the data from the given dataset (csv files).
* The fourth one is ***cosine\_algo.py***, which will perform the recommendation of the system using cosine similarity. It also includes the data-preprocessing, data cleaning functions along with the recommendation functions.

# 6. Implementation

## 6.1. Phase I – Method and Algorithm Discovery Phase

The method and algorithm discovery phase is the first phase of the project. In this phase, the research activities were conducted to determine which machine learning methods and algorithms will be used to implement of the movie recommendation system. Two activities were carried out during this phase. The first is selecting the information filtering method for the recommendation system, and the second is selecting the machine learning algorithm to assess its compatibility with the project's goals.

Based on the research and user’s requirements of the movie recommendation systems, the selected methods and algorithms were implemented in the Development Phase, Phase II. A detailed explanation of the selected machine learning methods can be found in [***Section 4: Methods and Algorithm Discovery.***](#_4.__)

## 6.2. Phase II – Development

After researching and selecting the machine learning methods for the movie recommendation system, the project was developed based on the users' requirements and design decisions. To complete the project within the allotted timeframe, the entire software development process follows the Agile methodology using SCRUM. The product backlog was broken down into two weeks for every sprint throughout the project. All the prioritized tasks were completed within the timeframe indicated. The product backlog items were prioritized whenever a new sprint started. The completed work was reviewed at the end of the sprint, and the progress report was submitted to the project advisor. Receiving feedback and submitting the project's progress report to the project advisor helped answer questions during the sprint retrospective, such as what went well? What went wrong? As a result, the sprint goals were met, and the project was completed successfully.

### 6.2.1 Project Structure Description

The directory of the project structure is as follows.

Graphical user interface, text

Description automatically generated

***Figure 6.1. The directory of The Project Structure***

1. **main.py *–*** the functions found in main.py is as follow.
   1. **recommender\_system ()**– this recommender function was imported from recommender.py and it will call the recommender system function.
2. **recommender.py** – the functions found in recommender.py are as follow.
   1. **recommender\_system ()** – This function acts like the user interface and it will let the user to choose different commands such as:
      * 1. Look up all movies
        2. Get a Recommendation of Similar Movies and
        3. Exit the Program.
   2. **get\_recommendations (user’s input)** – This function will call *get\_recommendation ()* function from cosine\_algo.py and display the results in console.
   3. **get\_recommend\_with\_add\_features (user’s input)** – This function will call *get\_recomend\_with\_add\_features ()* from cosine\_algo.py and display the results in console.
   4. **lookup\_all\_movies ()**– This function was imported from movie\_data\_report.py and it will display all the movies from the given dataset.
   5. **lookup\_title\_casts ()** – This function was imported from movie\_data\_report.py and it will display all the movie titles along with its casts from the given dataset.
3. **moive\_data\_report.py** – the functions found in this file are listed below.
   1. **get\_role (role, row)** -This function was used to create the new columns for the actor, director, writer and producer. These data were originally stored in crew columns and json files. Thus, it was easy to extract the data from crew column.
   2. **lookup\_all\_movie ()** – This function displays all movies from the dataset.
   3. **loolup\_title\_casts ()** – This function displays only title and cast columns from the dataset.
4. **cosine\_algo.py ()** – This file includes all the nltk and sklearn libraries and recommendation function. The functions found in this file are listed below.
   1. **preprocess\_sentences ()** – This function performs data preprocessing. Data preprocessing is the process of transforming raw data into the format that is understandable. Data pre-processing is essential for this movie recommendation system because the data needs to be clean, interpretable before we perform vectorization to cosine similarity.[9]
   2. **data\_cleaning ()** - This function will remove spaces and replaces the missing values for some of the data from the movie datasets. Since data cleaning is crucial in machine learning, the accuracy of the machine learning model will be jeopardy if the data is not clean.
   3. **get\_recommendations (title, cosine\_sim=consine\_sim) –** This function will perform movie recommendations by using cosine similarity. The function will call the functions, preprocess-sentences () and data cleaning () in order for the data (movie overview) to preprocess and clean. After preprocessing and cleaning data, vectorizing was performed using TfidfVectorize. After vectorization process, the cosine similarity was used to calculate the similarity score for the movies.
   4. **get\_recommendations\_with\_add\_features(title) –** This function will basically call get\_recommendation (title, cosine\_sim = consine\_sim) to calculate the similarity score of the moves with additional features.
5. **tests.py** – This file is under the directory*,* ***./tests*** which contains the following classes and functions.
   1. **TestRecommender (unittest.TestCase) –** This test case will test all the recommender system in **recommender.py.**
   2. **TestCosineAlgorithm (unittest. TestCase) –** This test case will test all the testable functions in **cosine\_algo.py.** Some of the functions were processing data and it would take so much time to test. Thus, only some of the functions were able to perform testing.
   3. **TestMovieDataReport(unittest.TestCase)** – This test case will test all the testable function from movies\_data\_report.py. Since **movies\_data\_report.py** is mainly used to ingest and process all the data, it’ll take so much time to test. Therefore, only some of the functions were able to perform testing in this file.
6. **Movie\_RS\_Data\_Visualization.ipynb –** The file is written in jupyter notebook and it was mainly used for data visualization during data analysis.

### 6.2.2 Development Results

This development results section will give the examples of the output of the recommendation system in console application and jupyter notebook.

**Example Result 1:** User Input: The Avengers

In this case, the user searched for the movie "The Avengers," and the recommendation system displayed the results in a Jupyter notebook.

Table

Description automatically generated

***Figure 6.2.1. Example of Getting Recommendation by Searching with Movie Title***

**Example Result 2:** User Input: Spirited Away, Daveigh Chase

***Figure 6.2.2***  depicts the result of the movie recommendation system when the user searches the title and cast of a movie. The user can also search for movies based on the title along with movie genres. Since our system uses the content-based filtering method, the system can recommend more accurate(similar) movies when a user searches the movie titles with additional features such as cast and genres.

Text

Description automatically generated

***Figure 6.2.2. Example of Getting Recommendation by Searching with Movie Title and Cast***

# 7. Operating Instructions

The project can be run locally.

* Using PIP
  + pip install -r requirements.txt
  + python run.py or execute main.py to start the program

After executing **main.py**, the following screen will show up in console.

Text

Description automatically generated

***Figure 7.1. User Interface When the Program is Started***

* Choose command 1) to see all the movies from the dataset.

Text

Description automatically generated

***Figure 7.2. Successfully display all the movies***

* Press 1) to see the moves with cast only. The system will display all the movies along with casts. The total numbers of rows and columns can be seen at the bottom of the data graph.

Graphical user interface, text

Description automatically generated

***Figure 7.3. Successfully display all the movies with casts***

* Press 2, to get a recommendation of similar movies.

Text

Description automatically generated with medium confidence

***Figure 7.4. Display Similar movies with Similarity Score Successfully***

* Press 3, to exit the system.

# 8. Recommendations for Enhancements

Since this is a simple movie recommendation model console line application, I would like to revise and improve this recommender system from the model’s and the website's perspectives in the future. The recommendation for enhancements of this project are as follows.

* GUI - The system will give a better user experience if the system has a user interface instead of just a command-line application.
* No Limitation for Users - User experience will be better if the system lets the user search for the movies without the current limitations( such as movie name or movie name with one additional feature or move name with two features). The user should be able to search based on his/her preferences.
* Optimize the data processing time - Optimizing the data processing time will improve the user experience.
* Web Application – the movie recommender should develop as a web application

# 9. Conclusion

Recommender systems are one of the most commonly used applications of machine learning technologies in today's business. They are an essential part of e-commerce and are used to promote their platforms. Our recommender provides higher accuracy of similar movies with additional features. Since our model recommends based on users' features, it can recommend similar movies to the user without needing to collect the user's past interaction with the system. In that way, we can avoid the drawbacks of recommending only the most popular items, which usually occurs in the nearest neighbor's algorithm model.

While the main focus of this project is to develop a movie recommendation system using a machine learning model to identify the similarity of the movies, we also emphasized analyzing data for accurate movie recommendations.

Lastly, since our research and analyses were mainly intended to preprocess the data and recommend similar movies to the user with higher accuracy, we felt that we met the project goals, and the analysis done in the project was efficient and insightful for the project's purpose as a whole. Using the existing model can revise and quickly enhance the recommendation system both from the perspective of the model and the website's perspective in the future.

# 10 . Acknowledgement

I would like to thank my professor, Dr. Ning Chen, for his direction, inspiration, patience, and constructive feedback throughout the project. Your wide knowledge and thorough editing have been a tremendous assistance in ensuring my success with this project. In addition, I'd like to thank Professor Bing Cong for approving my project proposal last summer.

# 11. GitHub Repository

The project, including the project report and presentation, can be accessed on the GitHub repository at the following URL.

<https://github.com/acxciv/Movie_RS>

# 12. Bibliography

[1] Vatsal. (2022, February 21). Recommendation Systems explained. Medium. Retrieved March 16, 2022, from

https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed

[2] A.M. Acilar, A. Arslan, A collaborative filtering method based on Artificial Immune Network Exp Syst Appl, 36 (4) (2009), pp. 8324-8332

[3] Dey, V. (2021, August 24). *Collaborative filtering vs content-based filtering for Recommender Systems*. Analytics India Magazine. Retrieved May 14, 2022, from https://analyticsindiamag.com/collaborative-filtering-vs-content-based-filtering-for-recommender-systems/

[4] Rocca, B. (2019, June 12). *Introduction to recommender systems*. Medium. Retrieved May 1, 2022, from https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

[5] (TMDb), T. M. D. (2017, September 28). *TMDB 5000 movie dataset*. Kaggle. Retrieved May 3, 2022, from https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?select=tmdb\_5000\_movies.csv

[6] Prabhu. (2019, November 21). *Understanding NLP word embeddings - text vectorization*. Medium. Retrieved May 2, 2022, from https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f7223

[7] Chamblee, B. (2022, February 7). *What is cosine similarity? how to compare text and images in Python*. Medium. Retrieved May 3, 2022, from https://towardsdatascience.com/what-is-cosine-similarity-how-to-compare-text-and-images-in-python-d2bb6e411ef0

[8] *Sklearn.feature\_extraction.text.TfidfVectorizer*. scikit. (n.d.). Retrieved May 15, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html

[9] *Text preprocessing NLP: Text preprocessing in NLP with python codes*. Analytics Vidhya. (2021, June 29). Retrieved May 15, 2022, from https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/