****

**University of Information Technology**

**2023-2024 Academic Year**

**CS-8316 (Data Analysis and Management)**

**Team Project Report**

In Partial Fulfillment of the Requirement for CS-8316 Team Project

**Title**

**Movie Recommendation System using**

**Cosine Similarity and KNN**

**Submitted by**

**Zaw Ye Naung 4SE - 874**

**Sat Paing Thu 4SE - 959**

**Minn Khant Ko 4SE - 968**

**Kaung Khant Nyein 4SE - 1222**

**Htoo Hla Oo 4SE - 1366**

**Ei Khine Moe 4SE - 1456**

**Contents**

**Chapter Title**  **Page No.**

**Chapter 1 Introduction** 1

1.1 Abstract 1

1.2 Objectives 1

**Chapter 2 Theory Background** 2

2.1 Machine Learning 2

2.1.1 Supervised Learning 2

2.1.2 Unsupervised Learning 3

2.1.3 Reinforcement Learning 3

2.2 Cosine Similarity 4

2.3 *k*-nearest neighbors algorithm (KNN) 4

**Chapter 3** **Movie Recommendation System** 5

3.1 Introduction to movie recommendation system 5

3.2 Types of movie recommendation systems 5

3.2.1 Content-Based Filtering 5

3.2.2 Collaborative Filtering 6

3.2.3 Hybrid Systems 6

3.2.4 Popularity-Based Recommendation 6

3.2.5 Knowledge-Based Recommendation 6

**Chapter 4 System Architecture** 7

4.1 System Overview 7

4.2 Data Flow Diagram 7

**Chapter 5 Dataset** 8

5.1 Overview 8

5.2 Dataset Structure 8

5.3 Data Sources 8

**Chapter 6** **Implementation**  9

6.1 Implementation with python 9

6.2 Evaluation Results 11

**Conclusion** 14

**References** 15

**Chapter 1**

**Introduction**

**1.1 Abstract**

In this article, a project related to the creation of a movie recommendation system based on cosine similarity of movies and K-Nearest Neighbors’ (KNN) algorithm is presented. Users are presented with one movie title as input and then ten movies recommended on the basis of relatedness are provided. For the cosine similarity, it states how similar two movie titles are in terms of how they are expressed as vectors and for KNN, it locates the nearest neighbors which suggest movies according to the content. Although the system is able to produce potential recommendations for the end-users, an evaluation construct has not been designed because the extent of accuracy or satisfaction of the users can’t be established. The idea of this project is to create visible and operative content-based movie recommendations schemes to users and add more enhancements to them such as depth of evaluation and user engagement feedback systems.

**1.2 Objectives**

The main objectives of this project are as follows:

- To design and implement a movie recommendation system using cosine similarity and KNN algorithms.

- To apply vectorization techniques for representing movie titles and measuring their similarity.

- To create a user-friendly interface that accepts an input movie title and generates a list of ten similar movies as recommendations.

- To explore the effectiveness of cosine similarity and KNN for movie recommendation purposes.

- To build a foundation for future enhancements, including the incorporation of performance evaluation metrics such as accuracy, precision, and recall, which are currently absent from the system.

- To examine potential scalability and real-world application of the system in online streaming platforms.

**Chapter 2**

**Theory Background**

**2.1 Machine Learning**

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data and thus perform tasks without explicit instructions. Recently, artificial neural networks have been able to surpass many previous approaches in performance.

ML finds application in many fields, including natural language processing, computer vision, speech recognition, email filtering, agriculture, and medicine. When applied to business problems, it is known under the name predictive analytics. Although not all machine learning is statistically based, computational statistics is an important source of the field's methods.

The mathematical foundations of ML are provided by mathematical optimization (mathematical programming) methods. Data mining is a related (parallel) field of study, focusing on exploratory data analysis (EDA) through unsupervised learning. From a theoretical viewpoint, probably approximately correct (PAC) learning provides a framework for describing machine learning. Machine learning approaches are traditionally divided into three broad categories, which correspond to learning paradigms, depending on the nature of the "signal" or "feedback" available to the learning system.

**2.1.1 Supervised Learning**

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data, known as training data, consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function allows the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Types of supervised-learning algorithms include active learning, classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. Examples of regression would be predicting the height of a person, or the future temperature.

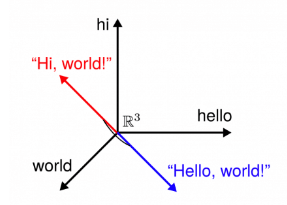
**2.1.2 Unsupervised Learning**

Unsupervised learning algorithms find structures in data that have not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. Central applications of unsupervised machine learning include clustering, dimensionality reduction, and density estimation. Unsupervised learning algorithms also streamlined the process of identifying large indel based haplotypes of a gene of interest from pan-genome. A special type of unsupervised learning called; self-supervised learning involves training a model by generating the supervisory signal from the data itself.

**2.1.3 Reinforcement Learning**

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In reinforcement learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcements learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

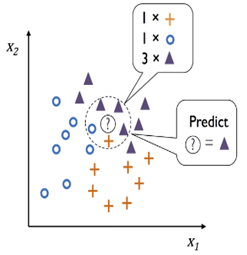
**2.2 Cosine Similarity**



(**Fig 2.1** Example Description of Cosine Similarity)

In data analysis, cosine similarity is a measure of similarity between two non-zero vectors defined in an inner product space. Cosine similarity is the cosine of the angle between the vectors; that is, it is the dot product of the vectors divided by the product of their lengths. It follows that the cosine similarity does not depend on the magnitudes of the vectors, but only on their angle. The cosine similarity always belongs to the interval [-1,1].

**2.3 *k*-nearest neighbors algorithm (KNN)**

****

(**Fig 2.2** Example Description of KNN)

In statistics, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric supervised learning method first developed by Evelyn Fix and Joseph Hodges in 1951, and later expanded by Thomas Cover. It is used for classification and regression. In both cases, the input consists of the *k* closest training examples in a data set. The output depends on whether *k*-NN is used for classification or regression.

**Chapter 3**

**Movie Recommendation System**

**3.1 Introduction to movie recommendation system**

A movie recommendation system, or a movie recommender system, is an [ML-based approach](https://labelyourdata.com/articles/achieving-digital-maturity-with-ai-in-business#machine_learning_and_business_use_cases) to filtering or predicting the users’ film preferences based on their past choices and behavior. It’s an [advanced filtration mechanism](https://labelyourdata.com/articles/how-to-choose-a-machine-learning-algorithm#supervised_ml_algorithms) that predicts the possible movie choices of the concerned user and their preferences towards a domain-specific item, aka movie. The basic concept behind a movie recommendation system is quite simple. In particular, there are two main elements in every recommender system: users and items. The system generates movie predictions for its users, while items are the movies themselves. The primary goal of movie recommendation systems is to filter and predict only those movies that a corresponding user is most likely to want to watch. The [ML algorithms](https://labelyourdata.com/articles/how-to-choose-a-machine-learning-algorithm#5_simple_steps_to_choose_the_best_machine_learning_algorithm_that_fits_your_ai_project_needs) for these recommendation systems use the data about this user from the system’s database. This data is used to predict the future behavior of the user concerned based on the information from the past.

**3.2 Types of movie recommendation systems**

Movie recommendation systems can be categorized into different types based on how they generate recommendations.

**3.2.1 Content-Based Filtering**

Content-based filtering recommends movies based on their characteristics (or "content") and the user’s previous interactions. This approach focuses on the attributes of movies (such as genre, actors, directors, or plot summaries) and compares them with the user's preferences. The system builds a profile for each user based on the movies they have rated highly or interacted with. It then recommends movies with similar attributes. If a user frequently watches action movies, the system will recommend other action movies with similar plots or themes.

**3.2.2 Collaborative Filtering**

Collaborative filtering leverages the behavior and preferences of multiple users to generate recommendations. This approach assumes that users who liked similar movies in the past will like similar movies in the future. Collaborative filtering analyzes patterns in user ratings or interactions. It identifies users with similar tastes and recommends movies based on what those similar users liked. If user A and user B have both enjoyed several sci-fi movies, and user B watches a new sci-fi film, that movie may be recommended to user A.

**3.2.3 Hybrid Systems**

Hybrid systems combine multiple recommendation techniques to overcome the limitations of individual methods. These systems often integrate content-based and collaborative filtering approaches to provide more accurate and diverse recommendations. A hybrid recommendation system may generate recommendations using both content-based filtering (based on movie attributes) and collaborative filtering (based on user preferences), then merge the results to offer a well-rounded list of movies.

**3.2.4 Popularity-Based Recommendation**

Popularity-based recommendation systems suggest movies based on their overall popularity, regardless of a user’s individual preferences. These systems recommend the most-watched or highly-rated movies to all users. The system ranks movies based on how many people have watched or rated them highly. It then recommends the most popular movies. A streaming platform may recommend a trending movie that’s being watched by millions of users.

**3.2.5 Knowledge-Based Recommendation**

Knowledge-based recommendation systems use explicit knowledge about users' preferences and movie attributes to make recommendations. These systems rely on pre-defined rules or queries rather than learning from user data. The system takes user-specified requirements (e.g., "recommend me a comedy movie directed by Woody Allen") and searches for movies that match the criteria.

**Chapter 4**

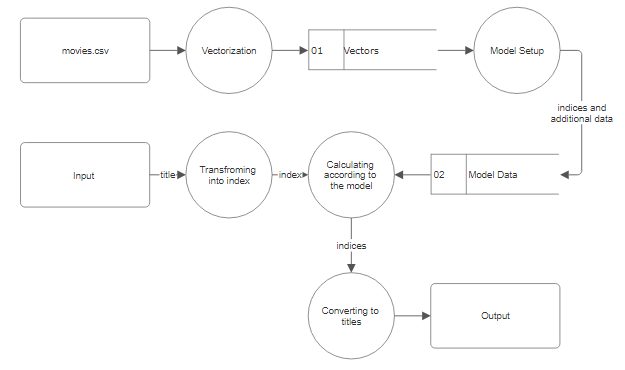
**System Architecture**

**4.1 System Overview**

The system architecture is divided into several core components, each responsible for different stages of data processing and recommendation generation. The key components include:

1. **User Input**: The system receives a movie title as input from the user.
2. **Vectorization & Indexing**: The input title and all other movie titles are transformed into vectorized formats to facilitate similarity calculations.
3. **Similarity Computation**: Depending on the selected algorithm (Cosine Similarity or KNN), the system computes the similarity between the input title and other movies.
4. **Recommendation Output**: The top 10 most similar movies are identified, converted back into movie titles, and displayed to the user.

**4.2 Data Flow Diagram**

****

(**Fig 4.1** Data Flow Diagram)

**Chapter 5**

**Dataset**

**5.1 Overview**

The dataset used for this movie recommendation system consists of a collection of 4,803 movies, each containing various attributes that describe its content, production details, and user ratings. These features provide a rich set of information that enables the recommendation system to generate personalized movie suggestions.

**5.2 Dataset Structure**

The dataset contains **24 columns**, representing various attributes of the movies. Some of the key attributes include:

1. **Title**: The name of the movie.
2. **Genres**: A list of genres associated with the movie (e.g., Action, Drama).
3. **Cast and Crew**: Information about the actors and the production crew involved in the movie.
4. **Director**: The director of the movie.
5. **Overview**: A short summary describing the plot of the movie.
6. **Popularity**: A numeric score reflecting the movie's popularity.
7. **Release Date**: The date when the movie was released.
8. **Vote Average and Vote Count**: The average user rating and the total number of votes the movie received.

**5.3 Data Sources**

The dataset was collected from publicly available movie databases, including information from platforms like **IMDb** and **TMDb**. It contains metadata about each movie, enabling a range of analytical approaches for generating recommendations.

**Chapter 6**

**Implementation**

**6.1 Implementation with python**

The first step is to import necessary libraries. We will use numpy for handling arrays, pandas for managing dataset, TfidfVectorizer for vectorizing, sklearn.metrics.pairwise for cosine similarity method and sklearn.neighbors for KNN method. The next step is to read the data from the dataset(.csv) and do data cleaning. Then combine the selected features for model training and transform them into vectors. Implementation are as below:

|  |
| --- |
| **movies\_data = pd.read\_csv('/content/movies.csv')**  **selected\_features = ['title','genres','cast','keywords','director','original\_language']**  **for feature in selected\_features:**  **movies\_data[feature] = movies\_data[feature].fillna('')**  **combined\_features = movies\_data['title']+' '+movies\_data['genres']+' '+movies\_data['cast']+' '+movies\_data['keywords']+' '+movies\_data['director']+' '+movies\_data['original\_language']**  **vectorizer = TfidfVectorizer()**  **feature\_vectors = vectorizer.fit\_transform(combined\_features)** |

For calculating with Cosine Similarity method, similarity for every movie is pre-calculated before input movie title is received. Input title is converted into index and compared with other movies indices for similarity, then they are sorted from most similar one to least similar ones. The movie titles are converted from each sorted index and printed back to the user. Implementation are as below:

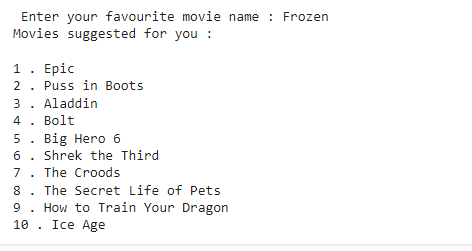
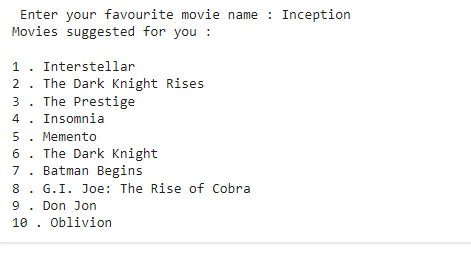
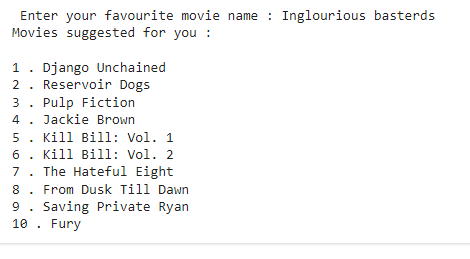
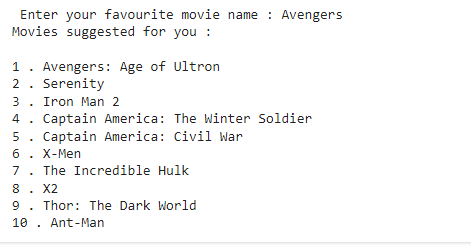
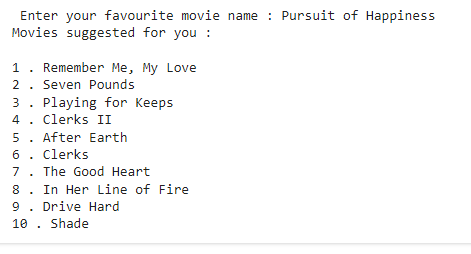
|  |
| --- |
| **similarity = cosine\_similarity(feature\_vectors)**  **movie\_name = input(' Enter your favourite movie name : ')**  **list\_of\_all\_titles = movies\_data['title'].tolist()**  **find\_close\_match = difflib.get\_close\_matches(movie\_name, list\_of\_all\_titles)**  **close\_match = find\_close\_match[0]**  **index\_of\_the\_movie = movies\_data[movies\_data.title == close\_match]['index'].values[0]**  **similarity\_score = list(enumerate(similarity[index\_of\_the\_movie]))**  **sorted\_similar\_movies = sorted(similarity\_score, key = lambda x:x[1], reverse = True)**  **sorted\_similar\_movies.pop(0)**  **print('Movies suggested for you : \n')**  **i = 1**  **for movie in sorted\_similar\_movies:**  **index = movie[0]**  **title\_from\_index = movies\_data[movies\_data.index==index]['title'].values[0]**  **if (i<11):**  **print(i, '.',title\_from\_index)**  **i+=1** |

For KNN method, the resulted movies are based on the distance between them and the input movie. This process is done at index level. Implementation are as below:

|  |
| --- |
| **knn\_model = NearestNeighbors(metric='cosine', algorithm='brute')**  **knn\_model.fit(feature\_vectors)**  **movie\_name = input(' Enter your favourite movie name : ')**  **list\_of\_all\_titles = movies\_data['title'].tolist()**  **find\_close\_match = difflib.get\_close\_matches(movie\_name, list\_of\_all\_titles)**  **close\_match = find\_close\_match[0]**  **index\_of\_the\_movie = movies\_data[movies\_data.title == close\_match]['index'].values[0]**  **distances, indices = knn\_model.kneighbors(feature\_vectors[index\_of\_the\_movie], n\_neighbors=11)**  **recommended\_movie\_indices = indices.flatten()[1:]**  **print('\n Movies suggested for you : \n')**  **for i, index in enumerate(recommended\_movie\_indices, start=1):**  **title\_from\_index = movies\_data[movies\_data.index == index]['title'].values[0]**  **print(i, '.',title\_from\_index)** |

**6.2 Evaluation Results**

The result for cosine similarity method and KNN methods and both same. The results for 5 movies are as follow:



Evaluating the accuracy of a recommendation system can be challenging since traditional metrics like accuracy (used in classification problems) may not apply directly. However, there are various approaches to assess the performance of a recommendation system, especially for movie recommendation systems using cosine similarity and KNN.

We will use a survey method among our classmates to check how our recommended movies are relevant. The accuracy for each movie is measured by rating among 0 to 10. The rating for each movie is as follow:

**Pursuit of Happiness**: Total votes - 34, Average rating - **5.6**

**Avengers**: Total votes - 34, Average rating - **9.5**

**Inglourious basterds**: Total votes -34, Average rating - **7.0**

**Inception**: Total votes - 34, Average rating - **5.3**

**Frozen**: Total votes - 34, Average rating - **7.8**

Total average rating is calculated through mean method:

Average Rating =

Average Rating =

Average Rating = **7.04**

Average rating for the system is **7.04** and the accuracy of the system is **70.4%**

**Conclusion**

In this project, we successfully designed and implemented a movie recommendation system utilizing two key algorithms: cosine similarity and K-nearest neighbors (KNN). The system takes an input movie title from the user and provides a list of 10 recommended movies based on the input.

The cosine similarity method allowed us to measure the textual similarity between movie descriptions, genres, and other metadata, helping to recommend movies that share similar content characteristics. The KNN algorithm, on the other hand, found the nearest neighbors in terms of feature vectors, making it effective in identifying related movies based on numerical relationships in the dataset. By combining these approaches, we created a robust recommendation engine capable of making accurate suggestions to users.

Throughout the development process, various techniques such as vectorization, data normalization, and feature extraction played a crucial role in preparing the data for both algorithms. However, the lack of ground truth data for evaluating the accuracy of the recommendations presented a challenge, leaving room for future improvements in the evaluation methods. Additionally, incorporating user feedback mechanisms and implementing metrics like precision, recall, and ranking-based evaluations could enhance the performance of the system.

This project highlights the importance of machine learning models and vectorization techniques in building recommendation systems that can personalize content based on user preferences. As future work, the system can be extended by including more sophisticated recommendation techniques such as matrix factorization, deep learning, or incorporating hybrid models that take into account user interaction data.

Overall, the project demonstrates how combining machine learning algorithms with well-structured datasets can create impactful real-world applications, such as personalized recommendation systems.

**References**

[**https://en.wikipedia.org/wiki/Machine\_learning**](https://en.wikipedia.org/wiki/Machine_learning)

[**https://en.wikipedia.org/wiki/Cosine\_similarity**](https://en.wikipedia.org/wiki/Cosine_similarity)

[**https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm**](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)

[**https://labelyourdata.com/articles/movie-recommendation-with-machine-learning**](https://labelyourdata.com/articles/movie-recommendation-with-machine-learning)

[**https://chatgpt.com/**](https://chatgpt.com/)