MOVIE RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

KASANUR JYOTHIRADHITHYA [RA2011003010416]
PRANISHA.R [RA2011003010427]
YESWANTH.J.B [RA2011003010432]

Under the guidance of

DR.ARUNA M

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degreeof

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **“MOVIE RECOMMENDATIONSYSTEM”** is the bonafide work of **KASANUR JYOTHIRADHITHYA [RA2011003010416] , PRANISHA.R [RA2011003010427] and YESWANTH.J.B [RA2011003010432]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein doesnot form any other project report or dissertation on the basis of which a degree oraward was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	SIGNATURE
Dr. Aruna M	**Dr.M.Pushpalatha**
Assistant Professor	Head of the department
Department of Computing Technologies	Professor and Head
	Department of Computing Technologies

Department of Computer Science and Engineering

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course	**:** B.Tech in Computer Science and Engineering
Student Names	**:** KASANUR JYOTHIRADHITHYA, PRANISHA.R, YESWANTH.J.B
Registration Number:	RA2011003010416, RA2011003010427, RA2011003010432
Title of Work	**:** Movie recommendation system

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web,etc.)
- Given the sources of all pictures, data etc. that are not my own

- Not made any use of the report(s) or essay(s) of any other student(s) eitherpast or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T.V.Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the projectwork.

We are incredibly grateful to our Head of the Department, **Dr. Aruna M,** Assistant Professor, Department of Computer Science and Egineering, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor, **Dr. P.Velmurugan**, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. Aruna M**, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Department of Computer Science and Engineering staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

KASANUR JYOTHIRADHITHYA [RA2011003010416]
PRANISHA.R [RA2011003010427]
YESWANTH.J.B [RA2011003010432]

ABSTRACT

A movie recommendation is important in our social life due to its strengthin providing enhanced entertainment. Such a system can suggest a set of movies to users based on their interest, or the popularities of the movies. Although, a set of movie recommendation systems have been proposed, most of these either cannot recommend a movie to the existing users efficiently or to a new user by any means. In this paper we propose a movie recommendation system that has the ability to recommend movies to a new user as well as the others. It mines movie databases to collect allthe important information, such as, popularity and attractiveness, requiredfor recommendation. It generates movie swarms not only convenient for movie producer to plan a new movie but also useful for movie recommendation. Experimental studies on the real data reveal the efficiency and effectiveness of the proposed system.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

API	application program interface
IMDB	internet movie database
CF	Collaboratory filtering
SLR	systematic literature review

CHAPTER 1

INTRODUCTION

It can be difficult for a user to find the right films that suit his or her likes given the vast number of films that are available worldwide. Users have varying tastes in performers or films. Finding a way to filter out pointless films is crucial, as is finding a list of relevant films.

A movie recommendation system works by doing the aforementioned activities. Such a system is based on the successful use of recommendation algorithms in various fields, including novels, TV shows, jokes, and news items. One of the most significant studies in the field of digital television is this one.

Collaborative filtering (CF) and content-based filtering are the basic foundations of the most well-known recommendation systems.From a list of active users, CF initially attempts to automatically identify groupings of similar people. The correlation measure is used to calculate user similarities. Then, based on the opinions of the user groups, it makes recommendations to the user. Although CF is effective in many areas, it has drawbacks including sparsity and scalability.CF searches for comparable users based on user ratings. However, because so few films have ratings, it is incredibly challenging to identify such.

CHAPTER 2

LITERATURE SURVEY

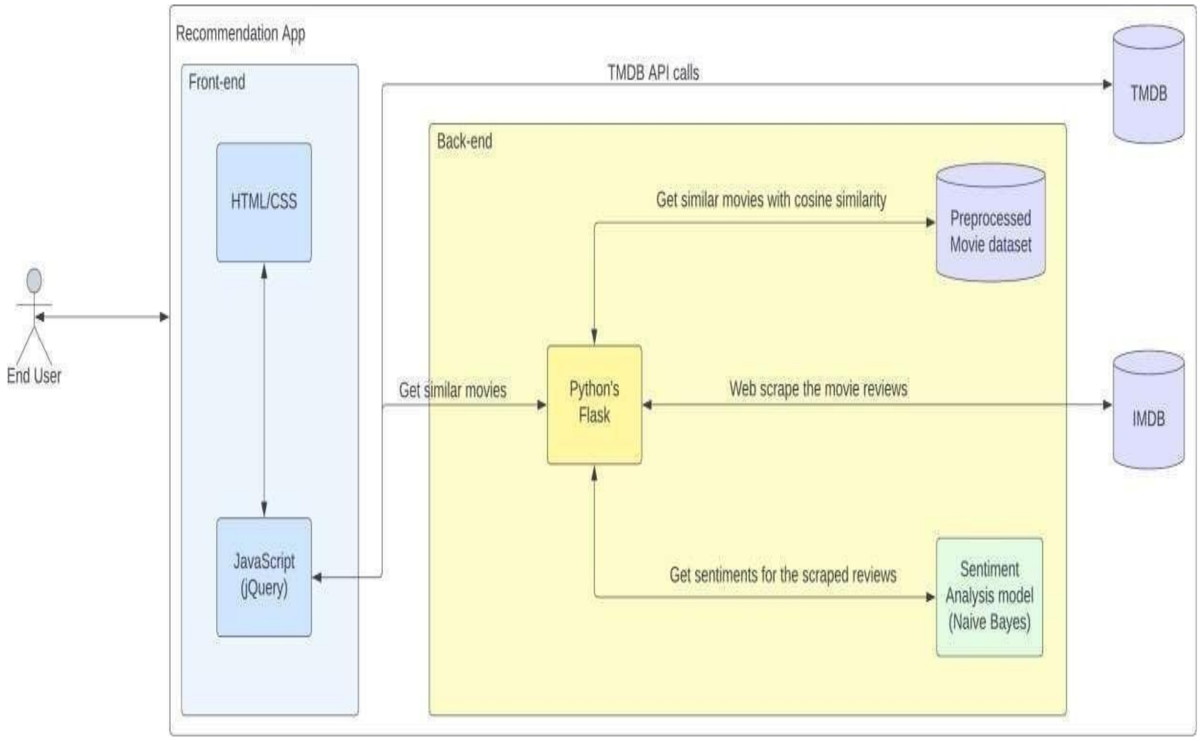
The information that is available on the internet is expanding quickly in the age of information and communication technologies. One of the technologies that is frequently used to filter information to handle the massive amount of information is recommender systems. Film is one of the emerging information. Movie streaming services like Netflix, Yiu, Disney Hotstar, and others have developed as a result of the growing number of films released each year. As a result, in order to facilitate and ensure that consumers have a positive experience using these services, technology for movie recommender systems is required. This study's objective is to conduct a systematic literature review (SLR) to compare different approaches to the algorithm created when creating a movie recommendation system.

Three stages make up the SLR method: planning, conducting, and reporting procedures. We considered studies published between 2010 and 2020. Out of the 21 major studies, 16 used collaborative filtering, 2 used knowledge-based filtering, and 3 used hybrid filtering. There were a total of 21 major studies. Findings from the SLR process show that each approach utilised to develop the movie recommendation system has advantages and disadvantages.

The model-based collaborative filtering approach is one tactic that helps minimise cold start, data sparsity, and scalability difficulties.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN


Recommendation App
Front-end
HTML/CSS
JavaScript (jQuery)
TMDB API calls
TMDB
Back-end
Get similar movies with cosine similarity
Preprocessed Movie dataset
End User
Get similar movies
Python's Flask
Web scrape the movie reviews
IMDB
Get sentiments for the scraped reviews
Sentiment Analysis model (Naive Bayes)

TECHNIQUE USED

Cosine similarity is a metric used in data analysis to compare two numerical sequences. Regardless of the size of the documents, cosine similarity is a statistic used to determine how similar they are. It calculates the cosine of the angle formed by two vectors that are projected onto a multidimensional space. The cosine similarity is useful since it increases the likelihood that the two comparable documents will be oriented closer together, even if they are separated by a large Euclidean distance because of the size of the documents. The cosine similarity increases with decreasing angle.

For instance, each word is given a unique coordinate in information retrieval and text mining, and a document is represented by a vector of the number of times each word appears in the document. Then, independent of the length of the papers, cosine similarity provides a good indicator of how similar two documents are likely to be in terms of their subject matter.

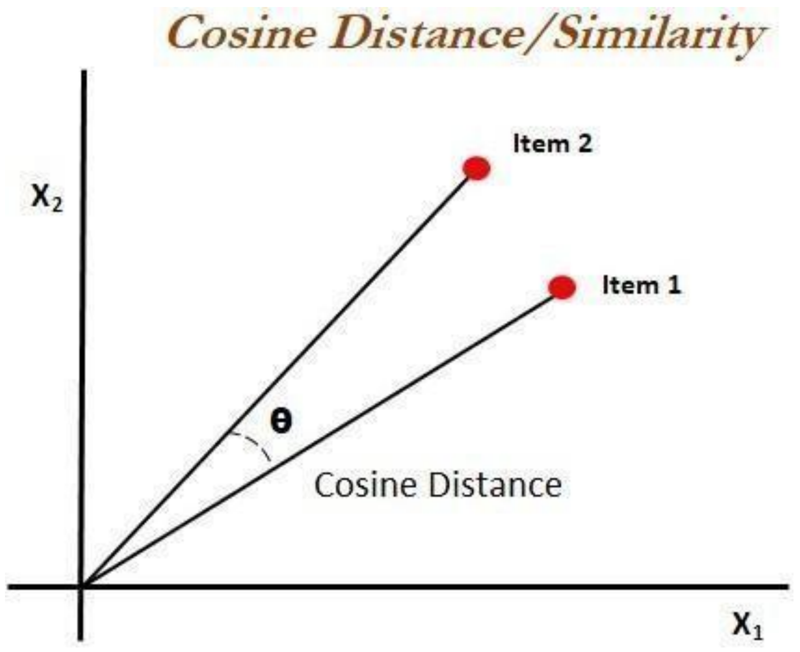
CHAPTER 4

METHODOLOGY

SIMILARITY SCORE

It is a numerical value that runs from 0 to 1, used to assess how much two items resemble one another on a scale from 0 to 1. This similarity score was calculated by comparing the text details of the two items. Therefore, the similarity score is a metric used to compare two things' given textual descriptions. Cosine-similarity can be used to achieve this.

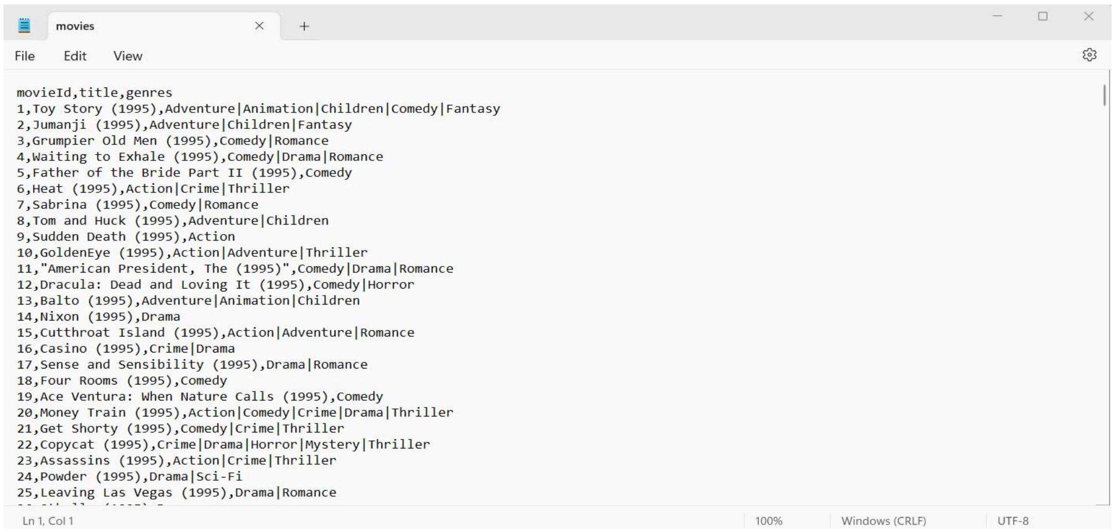
Regardless of the size of the documents, cosine similarity is a statistic used to determine how similar they are.
It calculates the cosine of the angle formed by two vectors that are projected onto a multidimensional space. The benefit of the cosine similarity is that it increases the likelihood that two similar documents will be orientated closer together, even though they are far apart by the Euclidean distance (because of the size of the documents). The cosine similarity increases with decreasing angle.



CHAPTER 5

DATASET

- **MovieLens Dataset:** One of the most popular datasets for movie recommendation systems is the MovieLens dataset. Along with movie metadata like title, genre, and year of release, it includes user ratings for films on a scale of 1 to 5. There are many sizes available, ranging from 100,000 to 25 million ratings.
- **IMDB Dataset:** The Internet Movie Database (IMDB) is a well-known website that offers details on motion pictures, television programmes, and celebrities. The IMDB dataset includes details about films, including its title, genre, star cast, and rating.
- **Kaggle Dataset:** 100K data points from different films and users are included in this collection.
- **Flixster Dataset:** Flixster is a website that offers reviews, ratings, and information on films. The Flixster dataset includes details about films, including their title, genre, star cast, and user ratings and reviews.
- **Yahoo! Movies Dataset:** This dataset includes user ratings and reviews along with details on movies like title, genre, rating, and cast.
- **The Movies Dataset:** This collection includes details about movies, including their runtime, budget, and cast in addition to other metadata like genre, rating, and title. Additionally, there are links to movie posters and trailers.



```
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
6,Heat (1995),Action|Crime|Thriller
7,Sabrina (1995),Comedy|Romance
8,Tom and Huck (1995),Adventure|Children
9,Sudden Death (1995),Action
10,GoldenEye (1995),Action|Adventure|Thriller
11,"American President, The (1995)",Comedy|Drama|Romance
12,Dracula: Dead and Loving It (1995),Comedy|Horror
13,Balto (1995),Adventure|Animation|Children
14,Nixon (1995),Drama
15,Cutthroat Island (1995),Action|Adventure|Romance
16,Casino (1995),Crime|Drama
17,Sense and Sensibility (1995),Drama|Romance
18,Four Rooms (1995),Comedy
19,Ace Ventura: When Nature Calls (1995),Comedy
20,Money Train (1995),Action|Comedy|Crime|Drama|Thriller
21,Get Shorty (1995),Comedy|Crime|Thriller
22,Copycat (1995),Crime|Drama|Horror|Mystery|Thriller
23,Assassins (1995),Action|Crime|Thriller
24,Powder (1995),Drama|Sci-Fi
25,Leaving Las Vegas (1995),Drama|Romance
```

CHAPTER 6

CODING AND TESTING

Index.html :

```
<!doctype html>
<html lang="en">
<head>

   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <meta name="description" content="Movie Recommender System">
   <meta name="author" content="Martin Kondor">

   <title>Movie Recommender</title>

   <!-- Icons -->
   <link rel="icon" sizes="115x93" href="https://martinkondor.github.io/img/icon.bmp">
   <link rel="shortcut icon" sizes="115x93" href="https://martinkondor.github.io/img/icon.bmp">
   <link rel="apple-touch-icon" sizes="115x93" href="https://martinkondor.github.io/img/icon.bmp">
   <!-- / Icons -->

   <!-- Styles -->
   <link rel="stylesheet" href="public/css/libs/bootstrap-4.3.1.min.css">
   <link href="public/css/main.css?v=2022062" rel="stylesheet">
   <!-- / Styles -->

</head>
<body class="bg">

   <div id="loaderSign" class="text-center">
      <div class="spinner-border text-white" role="status" style="width: 5rem; height: 5rem;"></div>
      <br>
      <p class="font-weight-bold text-uppercase" style="font-size: 20px;">
         Loading...
      </p>
   </div>
   <div class="container-fluid text-center mt-1">
```

```
    <h1 class="font-weight-bold h2 mt-4 mb-4">
        <a href="/" style="text-decoration: none; color: white;">🎬 Movie Recommender</a>
      </h1>

      <div id="box">
        <input id="title" type="text" class="form-control" placeholder="Type in a movie's title you liked
watching (e. g. cars, inception, titanic)" required="">
        <input id="year" type="text" class="form-control mt-2" placeholder="In what year it was released?
(Optional)" minlength="4" style="display: none;">
        <button id="submit" type="submit" class="btnbtn-primary mt-2 btn-block text-
uppercase">Recommend Movies!</button>
        <hr>

        <div id="box-list" class="mt-4">

          <p id="emptyText" class="text-muted mt-5">
            <i>
              The movies recommended for you will appear here...
            </i>
          </p>

          <h3 id="didYouMeanTitle" class="font-weight-bold mb-4">
            <div class="text-uppercase">
              Did you mean ...?
            </div>
            <p class="h5 text-muted">
              Choose the movie you've watched below.
            </p>
          </h3>

          <h3 id="weFoundNothing" class="text-uppercase font-weight-bold mb-4">
            We Found No Movies With This Title
          </h3>

          <h4 id="choosenMovie" class="mb-4">
          </h4>

          <div id="movie-list">
            <!--
            <div class="movie-li">
              1. Movie title
            </div>
            <div class="movie-li">
              2. Movie title
            </div>
```

```
                <div class="movie-li">
                    3. Movie title
                 </div>
                 -->
              </div>

           </div>

           <footer class="footer mt-5">
              <p class="text-muted" style="font-size: 12px;">
                 This project is made by <a href="https://MartinKondor.github.io/">Martin Kondor</a>,
                 see It's <a href="https://github.com/MartinKondor/MovieRecommender">GitHub</a> page
for more information.
                 <br>
                 Copyright &copy; Martin Kondor 2022
              </p>
           </footer>
        </div>
   </div>
   <!-- Scripts -->
   <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
   <script src="https://cdn.jsdelivr.net/npm/danfojs@1.1.0/lib/bundle.min.js"></script>
   <script src="public/js/main.js?v=2022062"></script>
   <!-- / Scripts -->

</body>
</html>
```

Main.css :

```
@charset "UTF-8";

html,
body {
   margin: 0;
   height: 100%;
   width: 100%;
   overflow-x: hidden;
   color: #fff;
   background: url('../img/andreas-gabler-XEW_Wd4240c-unsplash.jpg');
   background-size: cover;
   background-position: center;
}
```

```
label,
button {
    border-radius: 0px !important;
}

#box {
    width: 40%;
    min-height: 30%;
    padding: 40px 40px 5px 40px;
    box-shadow: 0px 5px 20px rgba(0, 0, 0, 0.222);
    border-radius: 10px;
    background-color: #fffffffb;
    margin: 0 auto;
    z-index: 1;
    color: #222222;
}
#weFoundNothing {
    display: none;
}
#didYouMeanTitle {
    display: none;
}

#choosenMovie {
    display: none;
}

#box-list {
    color: #000;
    text-shadow: none;
}

#loaderSign {
    position: absolute;
    margin: 0 auto;
    justify-content: center;
    justify-items: center;
    justify-self: center;
    background-color: rgba(0, 0, 0, 0.5);
    width: 100%;
    height: 100%;
    z-index: 2;
    padding-top: 150px;
}
```

```
.movie-li {
   margin: 20px 20px20px20px;
   border-style: dotted;
   padding: 20px;
   text-transform: uppercase;
}

.movie-li:hover {
   border-style: solid;
   cursor: pointer;
   transition: all 0.3s;
}

@media screen and (max-width: 1500px) {
   html,body {
      padding: 1px !important;
   }

   #box {
      width: 60% !important;
      padding: 20px 20px 5px 20px;
   }
}


@media screen and (max-width: 1200px) {
   html,body {
      padding: 0px !important;
   }

   #box {
      width: 95% !important;
   }
}


.bg {
   width: 100%;
   height: 100vh;
   display: flex;
   background-size: 300% 300%;
   background-image: linear-gradient(-45deg, #3493e6b2 0%, #EC6EADb2 100%);
   -webkit-animation: AnimateBG 10s ease infinite;
         animation: AnimateBG 10s ease infinite;
}
```

```
@-webkit-keyframes AnimateBG {
   0% {
      background-position: 0% 50%;
   }

   50% {
      background-position: 100% 50%;
   }
   100% {
      background-position: 0% 50%;
   }
}

@keyframes AnimateBG {
   0% {
      background-position: 0% 50%;
   }
   50% {
      background-position: 100% 50%;
   }
   100% {
      background-position: 0% 50%;
   }
}
```

Main.js :

```
'use strict';
(function ($) {
   // Prepare data
   function resetUI() {
      $("#emptyText").css("display", "none");
      $("#weFoundNothing").css("display", "none");
      $("#didYouMeanTitle").css("display", "none");
      $("#choosenMovie").css("display", "none");
      $("#loaderSign").hide();
      $("#movie-list").html("");
   }

   // Download data
   function load(callback) {
      resetUI();
      $("#loaderSign").show();
```

```
        $.getJSON("https://martinkondor.github.io/MovieRecommender/data/json/movies.json", function
(moviesData) {
            $.getJSON("https://martinkondor.github.io/MovieRecommender/data/json/ratings.json", function
(ratingsData) {
                let movies = new dfd.DataFrame(moviesData);
                let ratings = new dfd.DataFrame(ratingsData);
                let df = dfd.merge({ left: movies, right: ratings, on: ["movieId"], how: "inner"})
                callback(df);
            });
        });
    }

    function getMovieNameWords(title, callback) {
        let movieNameWords = [];
        let words = title.toLowerCase().trim().split(" ");
        for (let word of words) {
            if (word == "the") continue;
            movieNameWords.push(word);
        }
        callback(movieNameWords);
    }


    function findSimilarMovies(df, title) {
        let recommendedMovies = [];
        df.sortValues("rating", { inplace: true, ascending: false });

        // let movies = df.iloc({rows: ["0:5"]});
        // console.log(movies.values);

        // Get all ratings of the movie
        let ratings = [];
        for (let m of df.values) {
            if (m[1] != title) continue;
            ratings.push(m);
        }
        ratings = ratings.sort(function (a, b) {
            return a[4] < b[4];
        });

        // Get the top 5 ratings
        ratings = ratings.slice(0, 5);

        for (let rating of ratings) {
            let userid = rating[3];
            // Find movies rated by this user
            let userMovies = [];
            for (let m of df.values) {
                if (m[3] != userid || m[1] == title) continue;
                userMovies.push(m);
            }

            // Weight movies about the similarity of genres
            let scores = [];
            let genres = ratings[0][2].split("|");

            for (let m of userMovies) {
                let umGenres = m[2].split("|");
                let score = 0;

                for (let umg of umGenres) {
                    if (genres.includes(umg)) {
                        score += 1;
                    }
                }


                scores.push([m, score]);
            }


            // Sort by genre similiarity
            scores = scores.sort(function (a, b) {
                return a[1] < b[1];
            }).slice(0, 5);

            userMovies = [];
            for (let m of scores) {
                userMovies.push(m[0]);
            }

            userMovies = userMovies.sort(function (a, b) {
                return a[4] < b[4];
            }).slice(0, 5);

            for (let m of userMovies) {
                if (recommendedMovies.includes(m)) continue;
                recommendedMovies.push(m);
            }
        }
```

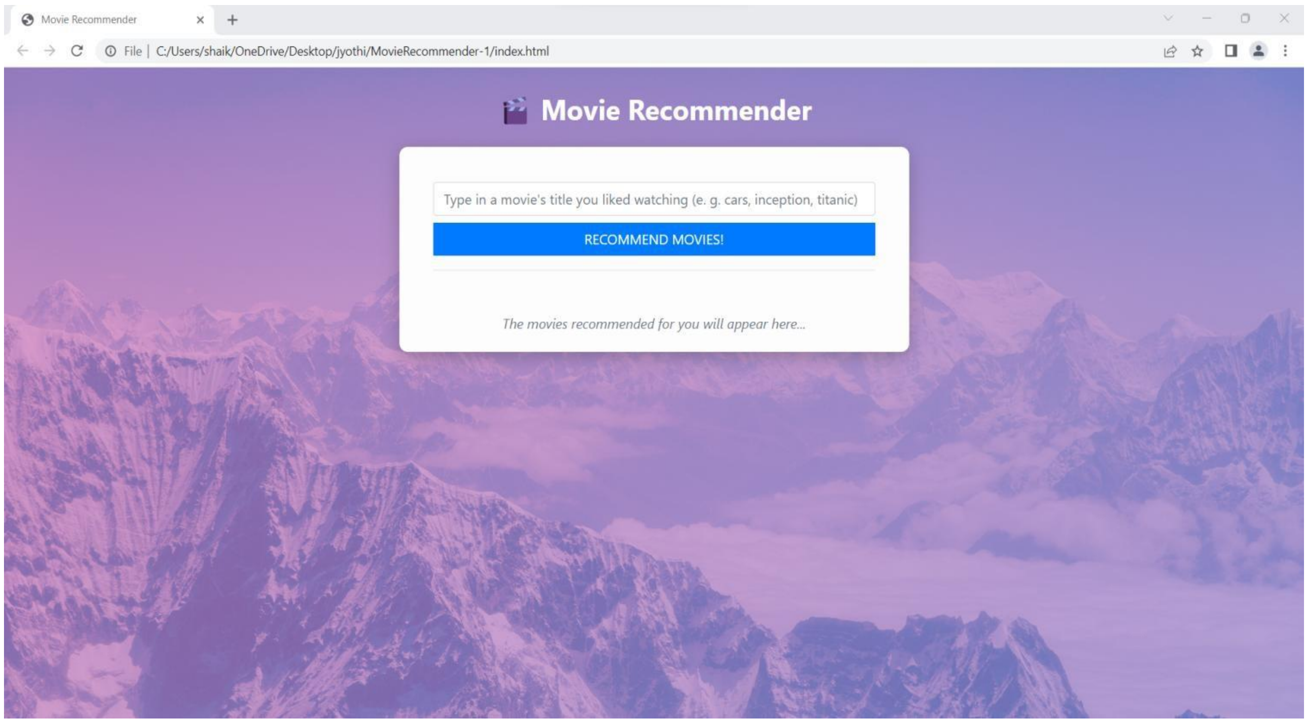
```
    // Adding recommendedMovies to html
    let alreadyUsedMovieTitles = [];
    let index = 0;

    for (let m of recommendedMovies) {
        if (alreadyUsedMovieTitles.includes(m[1])) continue;
        alreadyUsedMovieTitles.push(m[1]);

        $("#movie-list").html($("#movie-list").html() + `
            <div id="movie-${index}" class="movie-li" title="${m[1]}">
                ${m[1]}
            </div>
        `);

        // Recommend similar to the choosen movie
        // $("#movie-list").on("click", `#movie-${index}`, createMovieCallback(df, index, m[1]));
        $("#movie-list").on("click", `#movie-${index}`, function () {});  // Remove old listeners
        $(`#movie-${index}`).css("pointer-events", "none");
        $(`#movie-${index}`).css("cursor", "pointer");


        index++;
    }

    $("#title").val(title);
    $("#year").val("");
    $("#loaderSign").hide();
}

function createMovieCallback(df, index, title) {
    return function() {
        resetUI();
        $("#loaderSign").show();

        $("#choosenMovie").html("If you like <strong>" + title + "</strong> you might like these");
        $("#choosenMovie").css("display", "block");


        findSimilarMovies(df, title);
    }
}

function findMovie(df, title, callback) {
    getMovieNameWords(title, function (movieNameWords) {
        let similarTitles = [];
        let similarTitlesWithPoints = [];
```

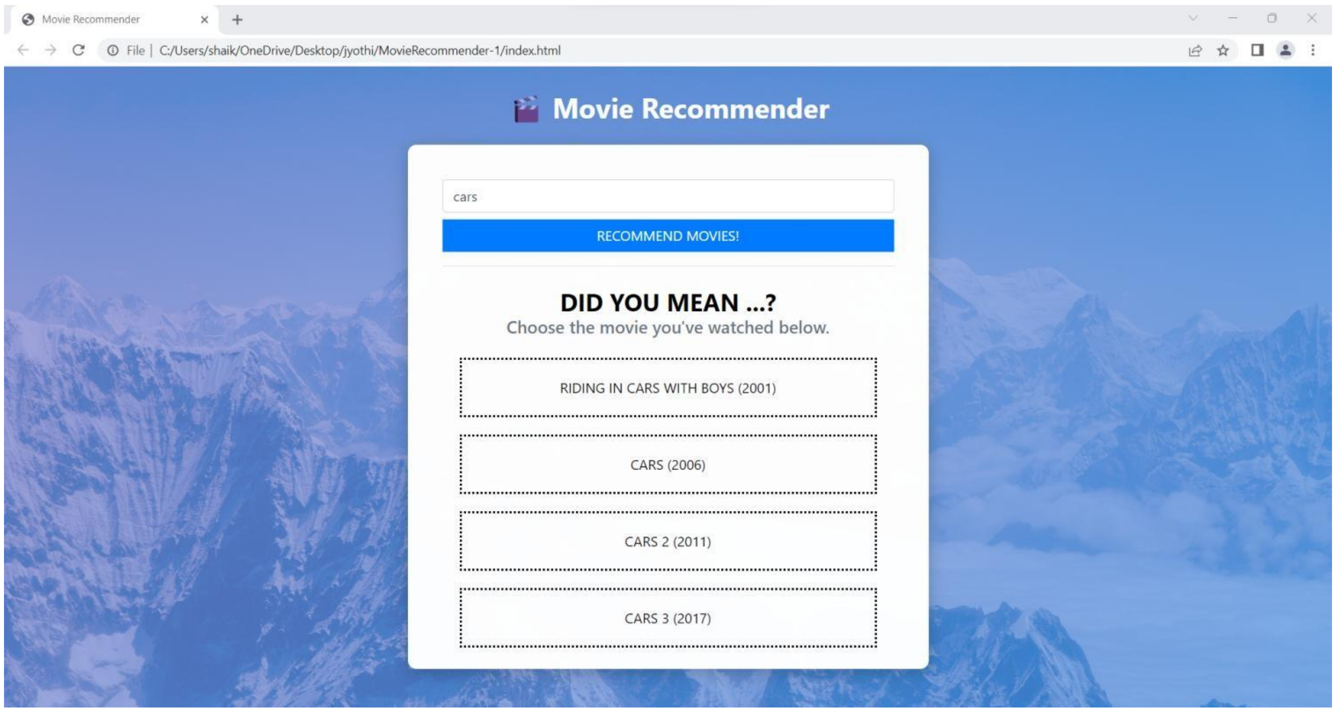
```
for (let dfTitle of df.iloc({columns: [1]}).values) {
    dfTitle = dfTitle[0];
    let wordsOfDfTitle = []
    let wordsOfDfTitleTemp = dfTitle.toLowerCase().trim().split(" ");

    for (let w of wordsOfDfTitleTemp) {
         if (w.trim() == "the") continue;
        if (w.trim()[0] == "(") continue;
        wordsOfDfTitle.push(w.trim());
    }

    let points = 0;

    for (let word of wordsOfDfTitle) {
        if (movieNameWords.includes(word)) {
            points += 1;
        }
    }


    if (points != 0 && !similarTitles.includes(dfTitle)) {
        similarTitles.push(dfTitle);
        similarTitlesWithPoints.push([dfTitle, points]);
    }
}

similarTitlesWithPoints = similarTitlesWithPoints.sort(function (a, b) {
    return a[1] < b[1];
});

if (similarTitlesWithPoints.length == 0) {
    $("#weFoundNothing").css("display", "block");
    return;
}

$("#didYouMeanTitle").css("display", "block");
let index = 0;

for (let m of similarTitlesWithPoints) {
    $("#movie-list").html($("#movie-list").html() + `
        <div id="movie-${index}" class="movie-li">
            ${m[0]}
        </div>
    `);
// On Click for each movie
            $("#movie-list").on("click", `#movie-${index}`, createMovieCallback(df, index, m[0]));
            index++;
         }

         callback();
      });
   }
   // / Prepare data

   function search(title) {
      load(function (df) {
         findMovie(df, title, function () {
            $("#loaderSign").hide();
         });
      });
   }

   function searchWithYear(title, year) {
      load(function (df) {

      });
   }

   function isYear(val){
      return !isNaN(val) &&val.length === 4;
   }

   $("#submit").on("click", function () {
      let title = $("#title").val();
      let year = $("#year").val() || "";

      // Running criterias
      if (title.length< 1) {
         /// TODO: Error messages
         return;
      }
      if (year.length != 0 && !isYear(year)) {
         /// TODO: Error messages
         return;
      }

      // Convert year if given to int
      let results = [];
      if (year.length != 0) {
         year = parseInt(year);
```

```
            results = searchWithYear(title, year);
        }
        else {
            results = search(title);
        }

        console.log(results);

    });

    resetUI();
    $("#emptyText").css("display", "block");

})(jQuery)
```

CHAPTER 7

SCREENSHOTS AND RESULTS


Movie Recommender
File | C:/Users/shaik/OneDrive/Desktop/jyothi/MovieRecommender-1/index.html
Movie Recommender
Type in a movie's title you liked watching (e. g. cars, inception, titanic)
RECOMMEND MOVIES!
The movies recommended for you will appear here...


Movie Recommender

File | C:/Users/shaik/OneDrive/Desktop/jyothi/MovieRecommender-1/index.html

🎬 Movie Recommender

cars

RECOMMEND MOVIES!

DID YOU MEAN ...?

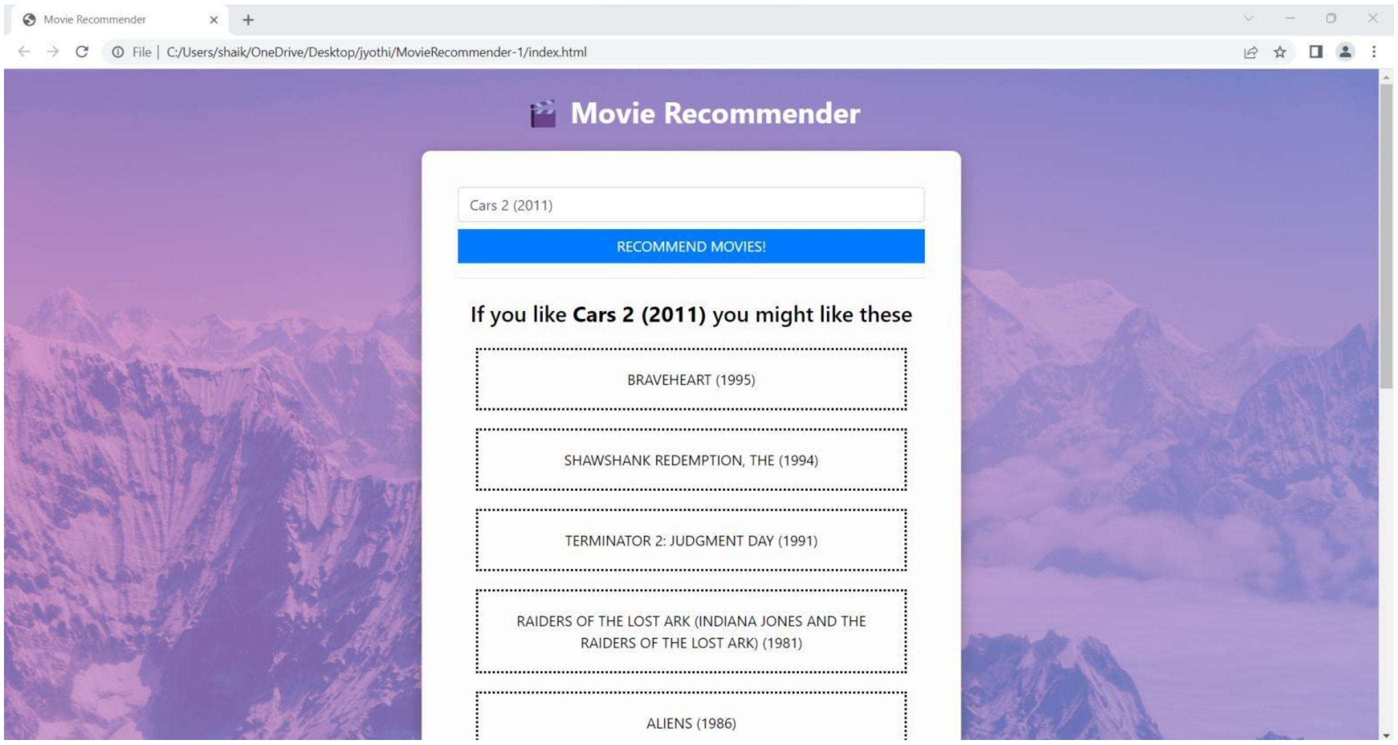
Choose the movie you've watched below.

RIDING IN CARS WITH BOYS (2001)

CARS (2006)

CARS 2 (2011)

CARS 3 (2017)


Movie Recommender

File | C:/Users/shaik/OneDrive/Desktop/jyothi/MovieRecommender-1/index.html

🎬 Movie Recommender

Cars 2 (2011)

RECOMMEND MOVIES!

If you like **Cars 2 (2011)** you might like these

BRAVEHEART (1995)

SHAWSHANK REDEMPTION, THE (1994)

TERMINATOR 2: JUDGMENT DAY (1991)

RAIDERS OF THE LOST ARK (INDIANA JONES AND THE RAIDERS OF THE LOST ARK) (1981)

ALIENS (1986)

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

The artificial intelligence technique of cosine similarity is used to create a movie recommendation system. In this case, data from many films and reviews are collected and analysed with the aid of APIs to determine how comparable the documents are, regardless of their size. Based on similarities and cast, recommendations are made. Based on our prior interests, we can use the movie recommendation system to suggest shows to watch.

REFERENCES

1. https://www.kaggle.com/

2. **Pratap Dangeti** "Statistics for Machine Learning [Book]"

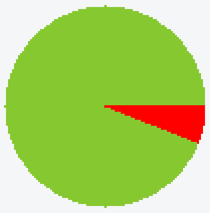
3. https://www.irjet.net/archives/V7/i9/IRJET-V7I9633.pdf

4. https://www.ijraset.com/research-paper/paper-on-movie-recommendation-system

5. M. Kumar, D. K. Yadav, A. Singh, and Y. Kr., International Journal Computer Applications, 7–11 (2015).

PLAGIARISM SCAN REPORT

Date	May 08, 2023
Exclude URL:	NO



Unique Content	94%
Plagiarized Content	6%
Paraphrased Plagiarism	0

Word Count	990
Records Found	4

CONTENT CHECKED FOR PLAGIARISM:

INTRODUCTION

It can be difficult for a user to find the right films that suit his or her likes given the vast number of films that are available worldwide. Users have varying tastes in performers or films. Finding a way to filter out pointless films is crucial, as is finding a list of relevant films.

A movie recommendation system works by doing the aforementioned activities. Such a system is based on the successful use of recommendation algorithms in various fields, including novels, TV shows, jokes, and news items. One of the most significant studies in the field of digital television is this one.

Collaborative filtering (CF) and content-based filtering are the basic small foundations of the well known recommendation systems. From a list of active users, CF initially attempts to automatically identify groupings of similar people. The correlation measure is used to calculate user similarities. Then, based on the opinions of the user groups, it makes recommendations to the user. Although CF is effective in many areas, it has drawbacks including sparsity and scalability.CF searches for comparable users based on user ratings. However, because so few films have ratings, it is incredibly challenging to identify such.

LITERATURE SURVEY

The information that is available on the internet is expanding quickly in the age of information and communication technologies. One of the technologies that is frequently used to filter information to handle the massive amount of information is recommender systems. Film is one of the emerging information. Movie streaming services like Netflix, Yiu, Disney Hotstar, and others have developed as a result of the growing number of films released each year. As a result, in order to facilitate and ensure that consumers have a positive experience using these services, technology for movie recommender systems is required. This study's objective is to conduct a systematic literature review (SLR) to compare different approaches to the algorithm created when creating a movie recommendation system.

Three stages make up the SLR method: planning, conducting, and reporting procedures. We considered studies published between 2010 and 2020. Out of the 21 major studies, 16 used collaborative filtering, 2 used knowledge-based filtering, and 3 used hybrid filtering. There were a total of 21 major studies. Findings from the SLR process show that each approach utilised to develop the movie recommendation system has advantages and disadvantages.

The model-based collaborative filtering approach is one tactic that helps minimise cold start, data sparsity, and scalability difficulties.

SYSTEM ARCHITECTURE AND DESIGN

TECHNIQUE USED

Cosine similarity is a metric used in data analysis to compare two numerical sequences. Regardless of the size of the documents, cosine similarity is a statistic used to determine how similar they are. It calculates the cosine of the angle formed by two vectors that are projected onto a multidimensional space. The cosine similarity is useful since it increases the likelihood that the two comparable documents will be oriented closer together, even if they are separated by a large Euclidean distance because of the size of the documents. The cosine similarity increases with decreasing angle.

For instance, each word is given a unique coordinate in information retrieval and text mining, and a document is represented by a vector of the number of times each word appears in the document. Then, independent of the length of the papers, cosine similarity provides a good indicator of how similar two documents are likely to be in terms of their subject matter.

METHODOLOGY

SIMILARITY SCORE

It is a numerical value that runs from 0 to 1, used to assess how much two items resemble one another on a scale from 0 to 1. This similarity score was calculated by comparing the text details of the two items. Therefore, the similarity score is a metric used to compare two things' given textual descriptions. Cosine-similarity can be used to achieve this.

Regardless of the size of the documents, cosine similarity is a statistic used to determine how similar they are.

It calculates the cosine of the angle formed by two vectors that are projected onto a multidimensional space. The benefit of the cosine similarity is that it increases the likelihood that two similar documents will be orientated closer together, even though they are far apart by the Euclidean distance (because of the size of the documents). The cosine similarity increases with decreasing angle.

DATASET

• MovieLens Dataset: One of the most popular datasets for movie recommendation systems is the MovieLens dataset. Along with movie metadata like title, genre, and year of release, it includes user ratings for films on a scale of 1 to 5. There are many sizes available, ranging from 100,000 to 25 million ratings.

• IMDB Dataset: The Internet Movie Database (IMDB) is a well-known website that offers details on motion pictures, television programmes, and celebrities. The IMDB dataset includes details about films, including its title, genre, star cast, and rating.

• Kaggle Dataset: 100K data points from different films and users are included in this collection.

• Flixster Dataset: Flixster is a website that offers reviews, ratings, and information on films. The Flixster dataset includes details about films, including their title, genre, star cast, and user ratings and reviews.

• Yahoo! Movies Dataset: This dataset includes user ratings and reviews along with details on movies like title, genre, rating, and cast.

• The Movies Dataset: This collection includes details about movies, including their runtime, budget, and cast in addition to other metadata like genre, rating, and title. Additionally, there are links to movie posters and trailers.

SCREENSHOTS AND RESULTS

CONCLUSION AND FUTURE ENHANCEMENTS

The artificial intelligence technique of cosine similarity is used to create a movie recommendation system. In this case, data from many films and reviews are collected and analysed with the aid of APIs to determine how comparable the documents are, regardless of their size. Based on similarities and cast, recommendations are made. Based on our prior interests, we can use the movie recommendation system to suggest shows to watch.

MATCHED SOURCES:

www.machinelearningplus.com - *1% Similar*Compare

https://www.machinelearningplus.com/nlp/cosine-similarity/

---

cottonandcloud.com - *<1>Compare*

https://cottonandcloud.com/amigurumi-cat-patterns/

---

www.researchgate.net - *<1>Compare*

https://www.researchgate.net/figure/complements-Figure-10-bu....

---

academic.oup.com - *<1>Compare*

https://academic.oup.com/innovateage/article/4/2/igz050/5687....
