

MOVIE RECOMMENDATION SYSTEM USING WORD EMBEDDING

**A Thesis Submitted
In Partial Fulfillment of the Requirements
for the Degree of**

MASTER OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING by

**AMIT SINGH
(23/CSE/01)**

**Under the supervision of
DR. RAJESH KUMAR YADAV**



Department of Computer Science and Engineering

**DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultpur, Main Bawana Road, Delhi-110042. India**

May, 2025

AKNOWLEDGEMENT

I have taken efforts in this thesis. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to **Dr. Rajesh Kumar Yadav** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing this thesis. I would like to express my gratitude towards the **Prof. Manoj Kumar (Computer Science and Engineering, Delhi Technological University)** for their kind cooperation and encouragement which helped me in the completion of this thesis. I would like to express my special gratitude and thanks to all the Computer Science and Engineering staff for giving me such attention and time.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete the thesis on time.



DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I, Amit Singh, Roll No. 23/CSE/01 student of M.Tech (Computer Science and Engineering), hereby certify that the work which is being presented in the thesis entitled “**Movie Recommendations System Using Word Embedding**” in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from August 2023 to Jun 2025 under the supervision of Dr. Rajesh Kumar Yadav, Associate Prof, Dept of Computer Science and Engineering. The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Place: Delhi

Candidate's Signature

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor



DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE

Certified that **Amit Singh** (Roll No. 23/CSE/01) has carried out the research work presented in the thesis titled “**Movie Recommendations System Using Word Embedding**”, for the award of Degree of Master of Technology from Department of Computer Science and Engineering, Delhi Technological University, Delhi under my supervision. The thesis embodies result of original work and studies are carried out by the student himself and the contents of the thesis do not form the basis for the award of any other degree for the candidate or submit else from the any other University/Institution.

Dr. Rajesh Kumar Yadav
(Supervisor)

Department of CSE
Delhi Technological University

Date:

ABSTRACT

As more movies are added online, it is becoming harder to decide what to watch. This project is designed to give movie recommendations, considering not only the keywords users type but also what those words truly mean. It employs Word2Vec to go over the metadata of films, including the names, genres and keywords and find interesting connections between them.

It was found that Word2Vec generated far better and more appropriate suggestions when evaluated against using CountVectorizer. The Skip-Gram model in Word2Vec was able to spot similar meaning between movies even when they didn't both contain the same words. The system learned from the TMDB dataset and its results were measured using Precision@10, Mean Reciprocal Rank (MRR) and NDCG. It was found that Word2Vec greatly improved the recommendations over strategies that depend on word frequency.

In general, the study finds that adding context-aware ideas makes movie recommendations more personal and meaningful to viewers. Improvements for the future could involve mixing this technique with collaborative filtering to improve both the accuracy and user enjoyment.

TABLE OF CONTENTS

Aknowledgement	ii
Candidate's Declaration	iii
Certificate	iv
Abstract	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
CHAPTER 1 INTRODUCTION	1-3
1.2. Problem Definition	2
1.3. Types of Recommendation System	2
CHAPTER 2 LITERATURE SURVEY	4-11
CHAPTER 3 SYSTEM ARCHITECTURE AND METHODOLOGY	12-25
3.1. Workflow	12
3.2. Dataset	13
3.3. Data Preprocessing	15
3.4. Methodology	16
3.4.1. Document Preparation	16
3.4.2. Word2Vec-Based Recommendation System	17
3.4.3. CountVectorizer-Based Recommendation System	19
3.4.4. Cosine Similarity Computation	20
3.5. Hardware Requirements	22
3.6. Software Requirements	22
3.7. Libraries	22
3.7.1. Python Software Ecosystem	22
3.7.2. Project Modules/Labraries	24
CHAPTER 4 EXPERIMENTAL RESULTS	26-28
4.1. Performance Evaluation Metric	26

4.2. Result.....	27
CHAPTER 5 CONCLUSION	29-29
5.1. Future Work	29
REFERENCES.....	30

LIST OF FIGURES

Fig. 1. Working Flowchart.....	12
Fig. 2. Distribution of Movie Genres.....	13
Fig. 3. Movie Release Trends Over Time	14
Fig. 4. Genre Evolution Over Decades.....	14
Fig. 5. Dataset after pre-processing	15
Fig. 6. Architecture of Word2Vec Skip-Gram Model.....	18
Fig. 7. Cosine Similarity.....	21
Fig. 8. Recommendations for "Thor" using CountVectorizer and Word2Vec SkipGram Models.	28
Fig. 9. Recommendations for "The Avengers " using CountVectorizer and Word2Vec SkipGram Models.	28

LIST OF TABLES

Table 1. Comparative analysis of movie recommendation system approaches highlighting models used, datasets, evaluation metrics, advantages, and limitations..	9
Table 2. Parameter of Word2Vec Skip-Gram Model, Bold text is decided after hyperparameter-tuning.	26
Table 3. Performance Metric after hyperparameter-tuning of the Word2Vec Model.	27

CHAPTER 1

INTRODUCTION

1.1. Overview

This technology exists as Recommendation System – A predictive system that predicts user preferences to generate item recommendations through mechanisms used by streaming platforms and e-commerce sites and social media and professional networks and search engines. Through recommendation engines users can discover various items such as movies and music as well as books and products all curated from their past actions. The post interaction activity on Facebook enables the platform to understand preferences in content consumption. The e-commerce platform uses its insight that during the rainy season raincoat purchasers also buy umbrellas to recommend this item to raincoat consumers.

Users in modern society require more than general categories which lack details. User engagement decreases whenever app expectations remain unmet which leads to dissatisfaction among users such as when a music streaming app produces unrelated songs to the user. Organizations have intensified their focus on smart recommender system development because of this reason. Many factors exist which create issues when enterprises aim to improve their work quality while developing logical and logistic systems. The preferences between users tend to differ permanently and temporarily for each parameter since user choices change with the day and season and also depend on weather as well as occupation circumstances.

Text similarity plays an indispensable part in information retrieval systems because it helps determine document relevance and supports other Natural Language Processing methods including recommendation systems and sentiment analysis and feature selection in text. Historically we depended on word frequency statistics that include Term Frequency-Inverse Document Frequency (TF-IDF) and CountVectorizer although sometimes deep semantic meanings of words fail to connect properly because the system frequently misses key information. Word2Vec possesses popularity as a deep learning model specifically for NLP tasks because it generates word vector spaces and maintains contextual word relationships.

The Google researchers introduced Word2Vec as their 2013 model which enables learning word representations efficiently. Word 2 Vectors presents a modern view of duty co-occurrence while surpassing traditional approaches such as TF-IDF and CountVectorizer and their co-occurrence methods. SkipGram proved to be an

effective method for predicting nearby terms from a target word thus aiding the understanding of semantic relationships. For several years Word2Vec remained a key influence on various present-day deep learning systems including FastText, GloVe and BERT because it enabled continuous advancement of word embedding methods. The recommendation system industrial standard consists of two primary sources namely the Internet Movie Database (IMDB) and Movie Database (TMDB). The metadata components found on these platforms extend to titles alongside genres keywords and cast members and crew names along with user assessment reviews of the movies. The contextual similarities between movies become easier to capture through Word2Vec thanks to this useful information during its model training process. TMDB provides an open API in addition to crowd-sourced data and operates under Amazon ownership with a database that drives streaming platform functionality.

The recommendation system development incorporates collaborative filtering with content-based filtering and deep learning model applications. The incorporation of Word2Vec among NLP techniques helps recommend additional movies to users through analysis of movie description alongside reviews and metadata.

1.2. Problem Definition

The problem that we are addressing in the project is the lack of an easy- to-use free and accurate movie recommendation system. The recommendation system we are proposing to build will be easily accessible to laymen without any requirement to subscribe to a pay-as- you-go system.

1.3. Types of Recommendation System

1.3.1. Content-Based Filtering

Content-based filtering analyzes item attributes through the evaluation of genre together with description in order to match items to user preferences. The algorithm recommends complementary films according to actors or genre preferences of customers who prefer action movies. Through previous user activity analysis this model creates a user profile from which it rates items according to their match with user preferences.

The method exhibits effectiveness when data appears scattered across systems since it does not depend on other users and concentrates on personalizing suggestions [14]. The filter bubble challenge exists because users encounter similar products while system requests complete metadata for each displayed item.

1.3.2. Collaborative Filtering

The output generation of collaborative filtering depends on user-item relationship data to generate recommendations. The methodology operates under two premises that users with matching preferences have maintained their consistent patterns thus predicting their upcoming preference behaviors. Collaborative Filtering is divided into:

User-Based Collaborative Filtering: This filtering suggests products which users enjoying similar preferences would appreciate. Two users become recommendation matches whenever they award the same movie with maximum points.

Item-Based Collaborative Filtering: The central goal in Item-Based Collaborative Filtering systems is to establish object similarities. The system guides users toward similar products through user rating analysis after they have dealt with a particular item.

The system provides benefits for pattern detection of user preferences but faces limitations due to its need for a big dataset to operate effectively and its sensitivity to new items and users [11].

1.3.3. Hybrid Approach

Several recommendation approaches unite in hybrid recommendation systems to generate precise and custom-made suggestions according to [8, 10 12, 13]. The solution merges content-based filtering with collaborative filtering into one system that solves their independent shortcomings. Netflix combines content-based filtering technology with collaborative filtering through their system that checks similar tastes between users. Word2Vec and Transformers among other deep learning models might be used sometimes to analyze complex relationships. Recommendations become more appropriate and varied and suited to individual user taste preferences through this dual recommendation methodology.

CHAPTER 2

LITERATURE SURVEY

Joseph and Nair [1] examine methods for making movie recommendations by applying CF and develop a content-based system. User-user, item-item and matrix factorization techniques are discussed, with the authors highlighting problems of data sparsity and cold-start. The researchers choose to use similarity techniques such as Cosine, Adjusted Cosine and Correlation, while discussing the influence of metrics such as Euclidean on their recommendations. The team uses the MovieLens data to conduct user-based and item-based CF through the use of Cosine similarity. The user-based model works better (RMSE 1.5 versus 2.5), as does the MAE (1.2 versus 2.2). Afterward, the team forms binarized data (genres and average rating) for films and they calculate the top recommendations for each user using a weighted mean. It accomplishes accuracy of ~80 percent with this system. They argue that CF needs a vast amount of ratings and content-based filtering doesn't perform well at suggesting unfamiliar goods. Authors encourage further work on using content features and clustering techniques in hybrid CF-content approaches to improve personalization, make recommendations faster and show less popular items.

Author Nousheen Taj et al. [2] introduce a method for personalized movie recommendations that appears in films with cosine similarity and uses combined categories of genre, actors, director and plot keywords in their approach. A Streamlit interface has been included so that users can add movies and download new updates of the recommendations. 4806 movies are processed by putting plot, keyword and genre data into TF-IDF vectors and measuring their similarity with cosine similarity to suggest the 10 most popular, most-voted or highest-rating similar movies. A toy evaluation involves finding cosine similarity and modifying the threshold to ensure recommendations are right 80% of the time. Divergences are addressed such as cold-starting, discrepancies in the quality of provided reviews/ratings and a reliance on which movies are most popular, with the authors emphasizing how the system keeps up-to-date on movie listings. They judge that cosine similarity is essential for delivering accurate recommendations and propose applying the approach to recommending books or medicines.

According to Sable et al. [3], a movie recommendation system that uses cosine similarity and sentiment analysis helps give personalized suggestions along with all the important details. From sources, TMDb and Wikipedia, the system collects metadata such as cast, genres, director, plot summary and release date, cleans and organizes it with Jupyter Notebooks and makes TF-IDF and Count Vectorizer representations for all movies. A movie title entered by the user prompts NLTK and a multinomial Naive Bayes classifier to assign scores of user opinions for

enhanced decision support. All the elements in the database are compared to the queried movie by calculating cosine similarity, then the top ten results are those that have the shortest Euclidean distance. The results are displayed on a web page, built with AJAX, HTML and JSON, so users can see ratings, when the show first aired and a basic summary of its reviews. The authors say that less work for humans and fast movie selection are clear benefits and propose future solutions such as location-based recommending, integrating other recommendation algorithms and regular database updates.

Ishika Naskar and Niju Joseph [4] describe a movie-recommending system that puts together content-based, demographic filtering and sentiment analysis to help make recommendations more personal and meaningful. They use a limited number of 4,890 MovieLens entries, match metadata like genres, keywords and cast data and generate “tag” vectors with CountVectorizer. Using cosine similarity, a movie selection engine will suggest titles that are like what a user likes and a demographic angle uses IMDb’s rating system to sort films by popularity. Movie reviews are obtained from IMDb, then processed (cleaned, broken down, stop-words removed), fired through a “ControX/Sen1” BERT model (which correctly sorts 93.8% of the times) and the outcome is summarized. Afterward, the movies in the first set are checked to select those with mostly good sentiments. The system gets a cosine similarity of 0.9931 between the sampled movies and their suggestions which means the recommendations are mostly aligned. Future improvements will cover adding real-time input, increasing the variety of data and looking into new types of hybrid network structures.

The authors [6] create a combined movie recommendation system using CB, Item-CF and User-CF methods to handle information overload and enhance personalization. This system is made up of a front-end, recommendation business, training using TensorFlow, data processing using Spark and live data handling with Flink so that both responsiveness and scaling are possible. The process of feature engineering applies correlation coefficients to contextual (time), item (genre, release date, rating) and user (rating history, demographic tags) statistics to compute similarities. By blending outputs from all three algorithms, ranking candidates according to the scenarios and displaying comparable and personalized movie lists, we can make personalized recommendations. In a qualitative report, 200 users gave satisfaction scores of 4 or 5 at 90%, while analysis reveals the hybrid approach achieved 81% accuracy and 70% coverage, finishing ahead of CB (72% accuracy, 62% coverage), Item-CF (76%, 65%) and User-CF (74%, 64%). Further, researchers will consider using deep learning in recommendations and ensure the system remains safe.

In [7], Sumathi et al. introduce a means of recommending movies using cosine similarity between different content elements and what users like. They use TMDB data on credits and movies, check the datasets for errors and combine cast, crew, genres, keywords, director, vote count and average to compute a new score.

Vectorization of features in texts is done by count vectorization, so we can use cosine similarity to compare movies. Using only movies that have a rating of 9 and higher, the system groups similar movies using details in their metadata. If you examine hypotheses for John Carter and Spider-Man 3, you'll notice the pattern. The system generates suggestions using what items are like, not based on what users are like, to ensure privacy is maintained. Prior to training, the authors describe how they removed common words, performed stemming and picked important features. According to the study, cosine similarity is effective in discovering movie similarities and recommending relevant shows and it explains how hybrid and updateable methods may be added for enhancement.

The model is introduced by Wei Zhao et al. [8] and is called LSIC. LSIC use adversarial training to combine long-term (MF) and short-term (RNN) data about movies and users. The generator uses features based on matrix factorization as well as dynamic LSTM information, while the discriminator in LSIC relies on a Siamese network to tell apart real high-rated movies from what is recommended. Cold-start problems are resolved by having the authors use ResNet-101 to encode movie posters, then directly input these visual features into the RNN at the beginning. A number of four hybrid approaches were investigated and LSIC-V4 attention weights the significance of both propagating and static factors more effectively than others. Using both the Netflix and MovieLens datasets, it is clear that LSIC outperforms earlier methods (like BPR, IRGAN, RNN) by up to 7.45% for Precision@5 and a high accuracy of 81% with 70% coverage. Future research includes trying to boost machine functions and look into how parts of the machine pay attention to each other.

Authors Sahu et al. [10] recommend a system with three modules meant to predict how well a movie will perform and who its main audience will be, during the initial phase of creating it. In the first module, I use genre, cast, director, keywords and description to compute TF-IDF vectors and cosine similarity to get the ten most similar movies from TMDb. In this module, ratings and votes for similar movies found on IMDb are used to develop a CNN that puts upcoming movies into six popularity classes. It records a significant 96.8% correct rate, making it better than traditional machine learning. The third module uses fuzzy c-means clustering on data for both age-wise voting and ratings to guess what each age group prefers, grouping them into junior, teenage, mid-age and senior groups. Thanks to the integrated setup, the process supports both early judgment calls and helps build content for particular viewers. In the near future, multimedia and analyses of emotions could be incorporated.

Mondal et al. [11] introduce a new movie recommendation system for Indian viewers that uses cross-attention methods. rather than focusing on written words and numbers, like ordinary methods do, this model analyzes audiovisual aspects from Hindi trailers and uses feedback from users noting if they like, dislike or feel neutral about the movies. Data from the Flickscore dataset is used to train the

system which consists of information about 510 Hindi films and details from more than 16,000 user-movie interactions. New innovations are the introduction of a GenreLike-score (GL-score) for comparing user genre preferences and movies and weighting user embeddings by how well their preferences fit with movie genres. We process audio with wav2vec2 and video is encoded using TimeSformer and keyframes selected equally. Multi-head cross-attention is applied to connect user and movie information to help predict what a user likes. Results show that the GL-score gives clearer accuracy, with audio outperforming video and liked-movie embeddings doing even better. Looking forward, researchers plan to enhance fusion methods, extend the variety of data and use technology that can identify emotions.

Kumar et al. [12] present a hybrid recommender system by combining CF, CBF and Twitter sentiment to increase how accurate the recommendations are. The system works with the MovieTweatings dataset which maps movie ratings to Twitter accounts and completes movie information using TMDb. We remove common words from tweets and standardize their text, then analyze them using VADER to score the sentiment of each movie. To fuse these scores, we combine metadata-based similarity (using a weighted model that follows social graph connections) and we apply a similar approach to the combined scores. The recommendation consists of a mix between the similarities of metadata and sentiment. The proposed approach is better in Precision@5 and Precision@10 than the baseline hybrid and sentiment-only models, obtaining scores of 2.54 and 4.97 respectively. The study demonstrates that a film's sentiment and IMDb rating are likely to move in the same direction. Future development will utilize information from multiple languages and formats and depend on emotion data to make better recommendations.

The authors [13] published an in-depth review highlighting RS that base recommendations on textual data such as reviews by users, articles, blogs and academic literature. From 2010 to 2020, the research sorts published literature into four main groups: datasets, ways to extract data, computational methods and ways to assess results. It notes that using TF-IDF, WordNet, ontologies, as well as Word2Vec, Doc2Vec are the key feature extraction ways. Because they capture semantic relationships well, word embeddings have become the technology most often used in the NLP field. The survey also investigates different computing methods, covering traditional metrics (cosine, Jaccard), machine learning algorithms and advanced deep models such as CNNs, RNNs and autoencoders, pointing out that shifting toward neural and hybrid systems can improve both how accurate and diverse the recommendations are. Additionally, the framework explains what precision, recall, F-measure and serendipity mean, as well as what coverage does. According to the paper, using hybrid and deep learning techniques is fast becoming necessary for efficient textual recommendations.

Mngomezulu and Ajoodha [14] explain that by using TF-IDF and RAKE to

extract movie keywords, their proposed method results in better recommendations than those made by classic collaborative filtering. From the MovieLens data, they get the movie plots and then use both extractors to make feature vectors that are used to compute similarity matrices. The system suggests 20 movies for each title entered and about half the time (7 out of 20), recommendations match up between the two methods. The highest percentage match (10) was seen for Casino, showing that both extractors could be used successfully together. It also uses the MovieLens 100K ratings dataset to help users make joint film choices. For the predictive task, the performances of SVD, KNN, SVD++ and CoClustering are checked with RMSE and MAE measures. I obtained the best results using SVD++ which had $RMSE = 0.90$ and $MAE = 0.71$. Results show that recommender systems work better if they rely on both keyword-based methods and collaborative filtering.

In their study, Izdihar et al. [17] build a movie recommendation system using Neo4j to examine connections between several attributes from the Netflix movies. The system puts movies, actors, directors, genres, countries and years into graph nodes, connecting them using edges labeled as ACTED_IN, DIRECTED and IN_CATEGORY. FastRP technology is used to produce low-dimensional vectors for each node that retain the structural resemblance between them. The system uses the embeddings to calculate how much two movies match, using the k-NN algorithm. With a perfect match score (1.0) for 270 films and thousands of others with similarity under 1.0, the method demonstrated how it can discover meaningful movie links. The analysis shows that mixing FastRP and k-NN inside Neo4j is a suitable way to make personalized recommendations and emphasizes the ability of graph architectures to scale and show context. It is also useful for making personalized updates and user preference models based on context.

Manwal et al. [18] introduce a movie recommendation system that compares movies by matching their textual features using TF-IDF and Bag-of-Words approach. The system uses a movie title as input and searches for similar films using cosine similarity which is more precise when there are lots of dimensions to consider. After preprocessing, the title, description and genre are made into vectors and compared to all other movie features. The similarity score serves to sort and present recommendations. Streamlit allows a web application interface to work, so that users can look for movies and see personalized recommendations, plus the posters of those movies. Because it depends on movie information and does not need users' historical behavior, the system resolves some major problems with collaborative filtering systems. They show that linking TF-IDF and BoW to movie plots improves the outcomes, resulting in better recommendations. Improvements can be achieved by mixing recommendation summaries with user behavior data to improve both accuracy and the ability to adjust over time.

Table 1. Comparative analysis of movie recommendation system approaches highlighting models used, datasets, evaluation metrics, advantages, and limitations.

Author & Year	Models Used	Performance Parameters	Datasets	Advantages	Disadvantages
Joseph & Nair (2022)	User-User & Item-Item CF, Content-Based	RMSE, MAE, Accuracy (~80%)	MovieLens	CF model comparison, content for cold-start	CF suffers from sparsity, lacks novelty
Nousheen Taj et al. (2024)	Content-Based (Cosine Similarity, TF-IDF)	Accuracy (~80%)	MovieLens (4806 movies)	UI with Streamlit, dynamic keyword filtering	Cold-start for users, popularity bias
Sable et al. (2021)	Content-Based + Sentiment (Naive Bayes, Cosine)	Not numerically detailed	TMDB, IMDb reviews	Sentiment integration, detailed UI	Limited metrics, dependent on review quality
Naskar & Joseph (2024)	Hybrid (Content + Demographic + Sentiment, BERT)	Cosine similarity (0.9931)	MovieLens, IMDb	Fuses metadata and sentiment	Complex system, lacks feedback loop
Pu & Hu (2023)	Hybrid (User-CF, Item-CF, CB, TensorFlow)	Accuracy (81%), Coverage (70%)	Not specified	Scalable, integrates deep learning	Complex to deploy, lacks cold-start focus
Sumathi et al. (2023)	Content-Based (Cosine Similarity, TF-IDF)	Qualitative recommendations	TMDB	Simple, easy to interpret	No performance metrics, lacks user data
Wei Zhao et al. (2020)	LSIC (Adversarial MF + RNN + Visual Embedding)	Precision@5 (+7.45%), Accuracy (81%)	Netflix, MovieLens	Cold-start via image, dynamic user modeling	High computational cost
Sahu et al. (2022)	Content-Based (TF-IDF + CNN + Fuzzy C-Means)	CNN Accuracy: 96.8%	TMDB, IMDb	Early-stage prediction, target audience classification	Not real-time, dependent on past trends
Mondal et al. (2024)	Multimodal (Audio/Video + Cross-Attention)	GL-score, qualitative	Flickscore	Indian audience focus, genre-sensitive modeling	Small dataset, lacks numerical accuracy
Kumar et al. (2020)	Hybrid (CF + CB + Twitter Sentiment)	Precision@5: 2.54, Precision@10: 4.97	MovieTweetings, TMDB, Twitter	Social media insights via sentiment	Sparse tweet coverage, no deep model

Kanwal et al. (2021)	Survey of Text-Based RS	Qualitative review	Various (2010–2020)	Comprehensive literature overview	No experiments or results
Mngomezulu & Ajoodha (2022)	Content-Based + CF (TF-IDF, RAKE, SVD, KNN)	RMSE (0.90–0.97), MAE (0.70–0.77)	MovieLens	Combines keyword extractors with CF	Limited keyword overlap, UI simplicity
Izdihar et al. (2024)	Graph-Based (Neo4j, FastRP, k-NN)	Similarity = 1 for 270 pairs	Netflix Movie Dataset	Graph model, interpretable links	No precision/recall, Netflix-specific
Manwal et al. (2023)	Content-Based (TF-IDF + BoW + Cosine Similarity)	Qualitative (via Streamlit)	Not explicitly named	Simple, user-friendly web deployment	No evaluation metrics, lacks hybrid modeling

A variety of approaches to developing movie recommendation systems are found in the reviewed research papers. Studies such as Joseph & Nair (2022) choose collaborative filtering based on user actions, but they usually meet problems due to the unavailability of user history or when few users have interacted on the website. Other researchers such as Nousheen Taj et al. (2024) and Sumathi et al. (2023), have examined content-based filtering by using the TF-IDF and cosine similarity methods. Such models are easy to use, but they are not strong enough to catch users' true preferences and tend not to show the measurable outcomes we require. Using a selection of techniques together, more advanced systems work more accurately and flexibly.

Likewise, Naskar & Joseph (2024) join demographic statistics and sentiment analysis via BERT and Pu & Hu (2023) use deep learning methods to generate recommendations for large groups. The reason these hybrid systems work better is that they are more difficult to operate and harder to manage. A few researchers rely on outside information such as reviews and discussion on social media. These papers (Sable et al., 2021 and Kumar et al., 2020) improve their recommendations with sentiment analysis, though this approach requires the text processed to be of a given quality and accessible.

Few research studies examine the latest possibilities. Wei Zhao et al. (2020) address the cold-start issue by building a deep learning model with visual representations and Mondal et al. (2024) work with Indian creators using text, images and videos, including cross-attention features. Izdihar et al. (2024) use Neo4j and a graph system to describe movie relationships, making insights flexible, but evaluation is not fully explored. To conclude, Kanwal et al. (2021) present text-based recommendation techniques and provide a summary of 10 years' worth of work, highlighting the major trends without including a novel model.

All in all, though basic content-based approaches are accessible and simple, hybrid methods and systems using deep learning are more powerful and flexible. Which approach to use depends on how simple the model needs to be, how right it is, how complex the systems are and how much data is available.

CHAPTER 3

SYSTEM ARCHITECTURE AND METHODOLOGY

3.1. Workflow

The process of machine learning model creation and appraisal becomes clear through Fig. 1 . The content-based movie recommendation system requires collecting a dataset which includes movie information about titles keywords genres and cast details. Before analysis the raw data requires processing to achieve proper organization and cleaning of information.

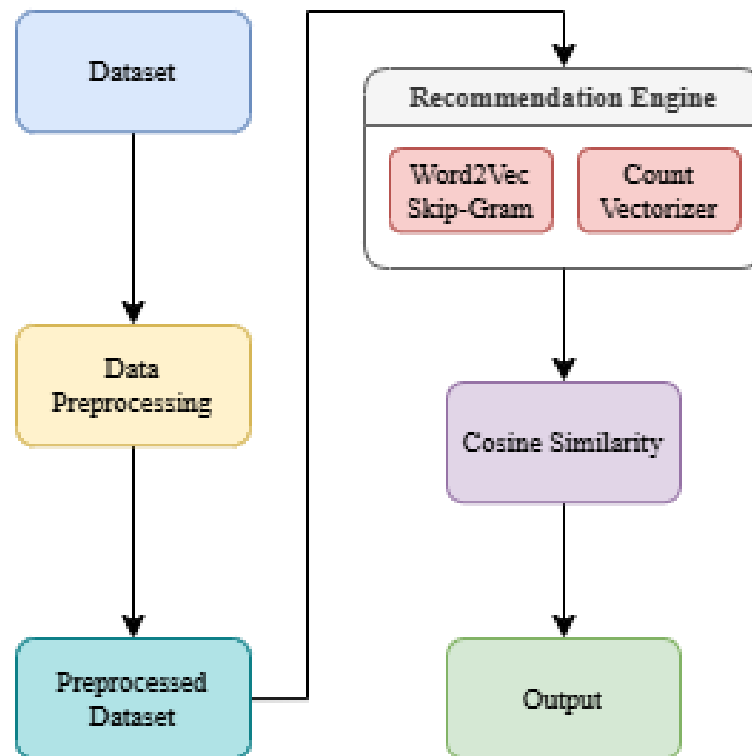


Fig. 1. Working Flowchart

The Count Vectorizer transforms textual data by generating a frequency-based matrix which displays token statistics for numerical representation. The pre-processed data consists of word embeddings together with token counts which serve as fundamental features for movie information extraction. The system utilizes feature vector cosine similarity calculation to evaluate the degree of movie similarity. The system yields recommended movies to users through their submitted movie titles by evaluating the calculated similarities in content. Users

receive individual recommendations of films with comparable themes or story elements after the system generates the suggestions. The recommendation system provides contextually appropriate results through Word2Vec and Count Vectorizer advanced techniques.

3.2. Dataset

By examining the TMDB 5000 dataset users gain access to extensive movie data since this dataset allows exploration of genre popularity trends as well as production durations together with thematic elements. A thorough pre-processing step enabled significant findings about cinematic trends to appear from the analysis of the dataset through exploratory data methods. A study of Fig. 2 demonstrates Drama takes the lead position among movie genres since it contains 2,000 entries while Comedy and Thriller rank as second and third. These genres position themselves at the top because issues and approach that appeal to everyone make them widely popular—Drama touches viewers with deep emotions and Comedy delivers laughter through joyous content. The specialized audience base of Westerns along with foreign releases and TV Movies explains the low number of films in each category. The data exhibits a wide array of film content that emphasizes the major positions of mainstream cinematic categories.

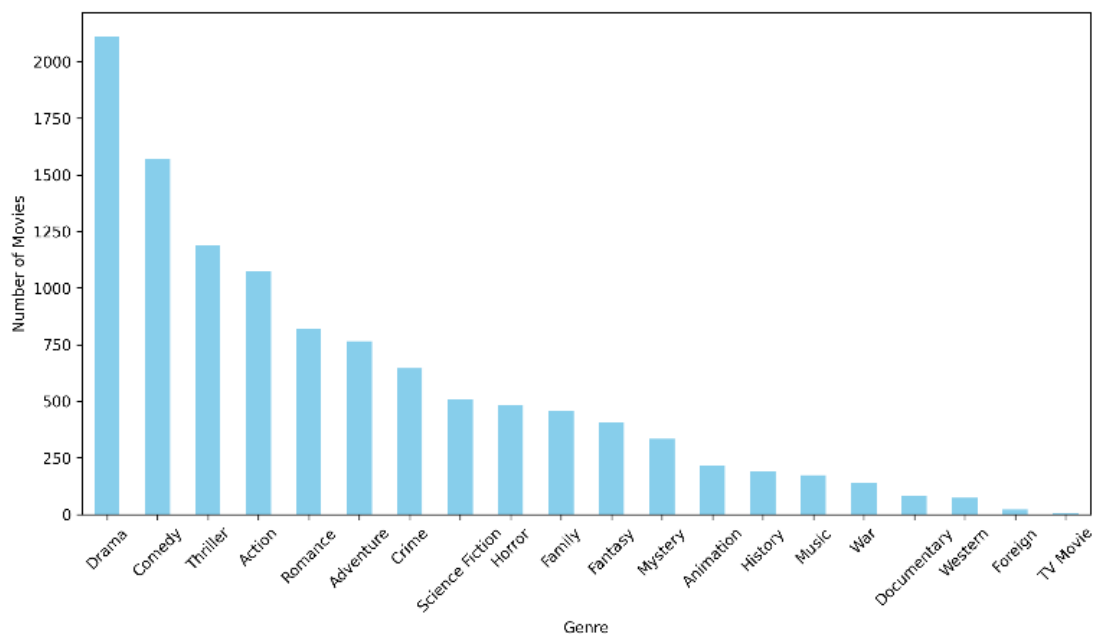


Fig. 2. Distribution of Movie Genres

Production levels steadily rose through time which reached its peak point in 2010 according to the histogram depicted in Fig. 3 . The pre-1960s period showed

modest movie releases because both technology and economics created obstacles. Movie production numbers surged dramatically throughout the post-1980s period because of better film technology together with cinema globalization and the development of blockbuster entertainment systems. A small drop seems to appear after 2016 yet it could be due to either missing data or balanced production trends. Technology advancements along with audience behavior patterns created the pattern through which the film industry grew.

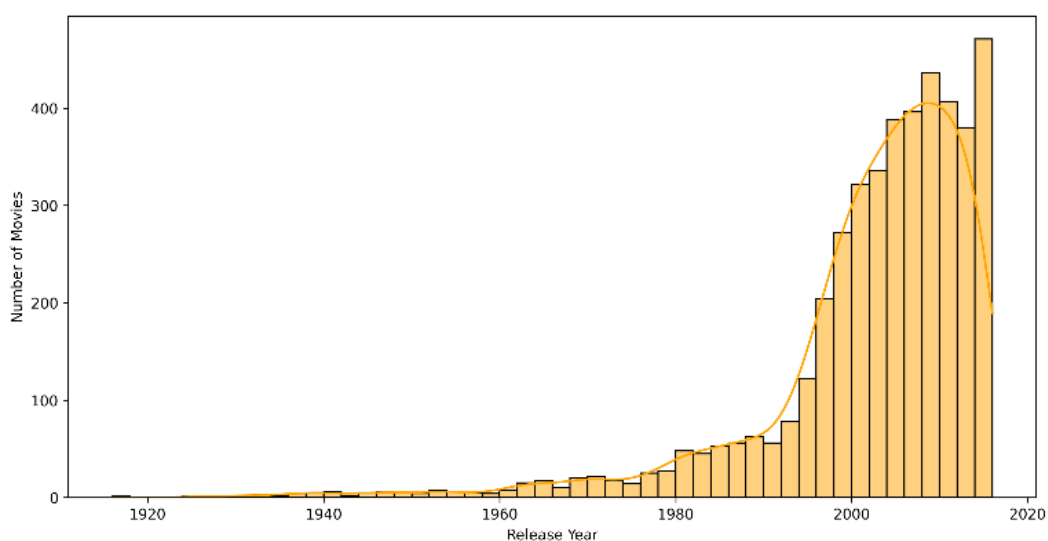


Fig. 3. Movie Release Trends Over Time

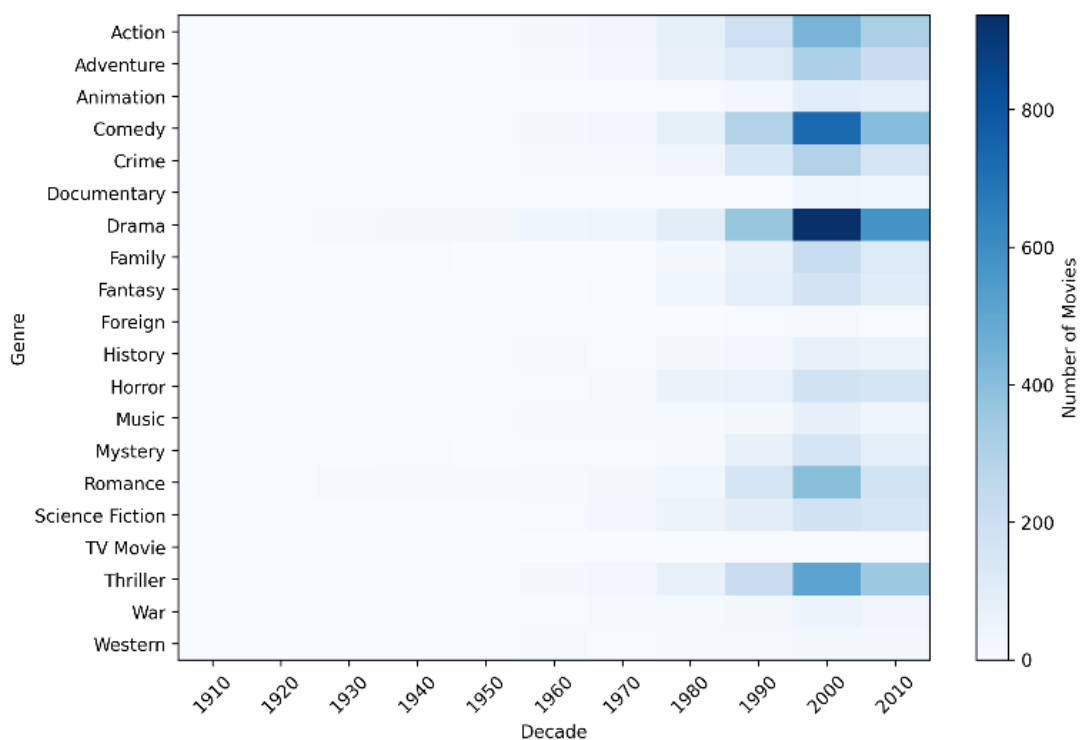


Fig. 4. Genre Evolution Over Decades

The research in Fig. 4 reveals interesting details about studio production throughout history stretching back to 100 years. Production counts for the Documentary genre along with other non-fiction films rose greatly during the 2000s as audiences became more interested in documentary artistic forms focusing on real-world matters. Drama and Comedy genres demonstrate robust consistency in their high numbers of yearly production because viewers consistently find them appealing. Popularity of Science Fiction and Fantasy movies during the 2000s and 2010s results from higher technology standards which fuel creative storytelling in film. The cultural interest in Westerns has decreased thus showing a decline in Western films numbers. Major war conflicts of World War II demonstrate societal changes through the heatmap's display of higher production numbers of War movies. This visual display demonstrates that movie industry trends emerge from constantly shifting production dynamics throughout changing popular fame genres.

3.3. Data Preprocessing

The TMDB dataset needed multiple processing operations that increased its suitability for use in building a recommendation system. The 'genres' and 'keywords' columns held JSON-like structures when the data was initially gathered. A simpler form of analysis became possible after genres received separate extraction from keywords then merged into combined comma-separated strings for distribution research. Such text processing capabilities make the Word2Vec model effective for topic analysis.

	0
title	Avatar
overview	In the 22nd century, a paraplegic Marine is di...
genre	Action,Adventure,Fantasy,Science Fiction
keywords	culture clash,future,space war,space colony,so...
cast	Sam Worthington,Zoe Saldana,Sigourney Weaver,S...
year	2009
combine_tk	avatar culture clash,future,space war,space co...

Fig. 5. Dataset after pre-processing

The team generated the 'combine_tk' column that assembled different text elements including titles and keywords into one unified text value. Word embeddings benefit from the combination elements to produce accurate content-based recommendations. A system of text normalization methods normalized all text data by lowering case while eliminating special characters and punctuation and removing stopwords from the text. Following tokenization the train-up phase for Word2Vec began.

The database received pre-processing that included subdivision of numerical and categorical elements while extracting the 'release_date' column values for time-based research purposes. Null values in essential columns probably underwent removal or imputation to keep the data quality high. Extra features that accounted for text lengths as well as keyword counts could have been considered to analyze movie complexity. During pre-processing the prepared dataset formed a collection of movie data which included the main components Title, Overview, Genre, Keywords, Cast, Release Year and Combined features composed of Title and Keywords data shown in Fig. 5 . The processing sequence prepares data for Word2Vec Skip-gram implementation by standardizing its text elements into reliable features with high information density. Through the integration of multiple text features the model receives training capabilities to understand important word embeddings that express movie structural relationships. The correct generation of content-based recommendations through cosine similarity depends heavily on these processes. The pre-processed dataset creates strong base conditions to examine genre development patterns and chronological patterns along with building complex recommendation systems for the field of movie analysis.

3.4. Methodology

This chapter details the design, implementation, and functioning of the two core components of the proposed movie recommendation system: one based on Word2Vec embeddings and the other using CountVectorizer. Both approaches fall under the domain of content-based filtering, where the goal is to recommend movies similar to a given movie based on its textual metadata (e.g., title, keywords).

3.4.1. Document Preparation

The first step common to both models is to prepare a clean and meaningful textual representation of each movie. This is achieved through the function `prepare_documents`, which performs the following tasks:

- **Combining Metadata:** Concatenates the movie title and associated keywords.

- **Normalization:** Converts text to lowercase.
- **Tokenization and Filtering:** Removes punctuation and filters out tokens shorter than two characters.
- **Reconstruction:** Joins the cleaned tokens back into a whitespace-separated string.

The result is a processed document for each movie that captures the semantic essence of its content.

3.4.2. Word2Vec-Based Recommendation System

Word2Vec functions as a neural network algorithm that develops dense word vectors although it specifically teaches models to interpret semantic concepts through contextual word relationships. Through the implementation of the Skip-Gram model users can predict surrounding words after providing a target word for prediction. The embeddings produced by Skip-Gram become optimized through a process which maximizes the probability of window-sized neighboring words against traditional frequency-based TF-IDF or CountVectorizer models.

Mathematically, the model optimizes the following objective function:

$$\max \sum_{w \in V} \sum_{c \in C(w)} \log P(c|w) \quad (1)$$

The model seeks to maximize the following target function where w stands for vocabulary word w from vocabulary V and $P(c|w)$ indicates the conditional probability of context word c when given target word w and $C(w)$ refers to the context words which appear near w within a defined window scope.

A two-layer neural network architecture within the model accepts one-hot encoded words from the input layer and passes them through hidden layers for reducing dimensions before producing output layer predictions of context words as depicted in Fig. 6 . The model processing efficiency rises because negative sampling enables the weight update of only selected fractions. The research uses a Skip-Gram model to process movie metadata featuring titles and keywords which generates meaningful word representations in embedded space. Semantically advanced associations become manifest in movie recommendations through this advanced method leading to superior results than traditional techniques.

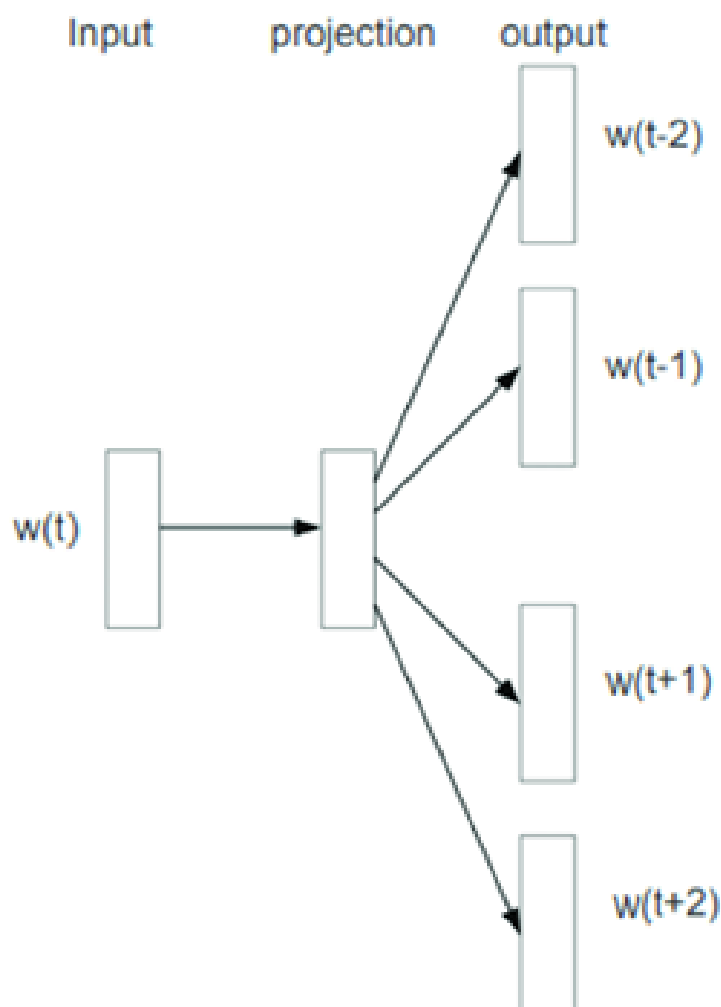


Fig. 6. Architecture of Word2Vec Skip-Gram Model

➤ Training Word2Vec Embeddings

The function `train_word2vec` takes the preprocessed documents and trains a Word2Vec model using the Skip-gram or CBOW algorithm. It supports tuning of the following hyperparameters:

- **vector_size:** Dimensionality of the word vectors (default 150)
- **window:** Context window size (default 3)
- **min_count:** Minimum word frequency threshold (default 2)
- **sg:** Skip-gram (1) or CBOW (0) mode

- **epochs:** Number of training iterations
- **negative:** Number of negative samples
- **sample:** Downsampling rate for frequent words

The model outputs:

- The trained Word2Vec model
- Averaged document vectors for each movie

➤ **Hyperparameter Tuning**

To optimize the performance of the Word2Vec embeddings, the function `tune_word2vec_hyperparams` performs a grid search over a specified parameter space. Each combination is evaluated using an appropriate metric (e.g., cosine similarity), and the best model is retained.

➤ **Generating Recommendations**

Once vector representations are generated, the function `recommend_similar_movies` calculates the cosine similarity between the target movie and all others. The top-N most similar movies are then recommended. This is implemented inside `getRecommendations_Word2Vec`, which integrates the above steps.

Advantages:

- Captures semantic similarity between words and contexts.
- Flexible and tunable.
- Performs well for sparse data.

Disadvantages:

- Training is resource-intensive.
- Sensitive to hyperparameter settings.

3.4.3. CountVectorizer-Based Recommendation System

The CountVectorizer model produces word-count matrices by implementing the established bag-of-words (BoW) method on text data. The method splits text documents into distinct words for building a common vocabulary and enables document comparison using word frequency patterns. CountVectorizer serves

basic text processing operations adequately while it experiences multiple limitations in its application. As a result of this approach word elements become independent units that lack collaborative semantic meaning between words. The transformation of high-dimensional sparse representations turns into an extensive issue since CountVectorizer lacks efficiency with big datasets. With CountVectorizer it becomes technologically impossible to study the relationships between words since the algorithm uses word occurrence counts to generate numerical vectors from movie descriptions. CountVectorizer delivers reliable first-stage solutions for text similarity tasks despite having well-documented constraints. This study compares CountVectorizer to Word2Vec Skip-Gram by scoring their output similarity along with an evaluation of their performance in generating movie recommendations.

➤ **Vectorization with CountVectorizer**

The function `get_countvectorizer_vectors` uses Scikit-learn's CountVectorizer to transform each document into a vector of token counts. Configurable parameters include:

- **max_features:** Limits vocabulary size
- **ngram_range:** Supports unigrams, bigrams, or higher

The output is a dense feature matrix and the fitted vectorizer.

➤ **Generating Recommendations**

Similar to the Word2Vec method, the function `recommend_similar_movies` computes cosine similarity between the vector of the input movie and all others. This logic is encapsulated in `getRecommendations_countVectorizer`.

Advantages:

- Easy to understand and implement.
- No training required.
- Fast and efficient.

Disadvantages:

- Ignores semantic similarity between different but related words.

Generates sparse, high-dimensional vectors.

3.4.4. Cosine Similarity Computation

The text representation similarity measurement uses Cosine similarity as its algorithm which receives broad acceptance among researchers. The value of cosine similarity derives from measuring the angle formed between two vectors

through mathematical calculation as shown in fig 7 . The calculated results span between 0 to 1 so that 0 represents non-similar vectors while 1 stands for exactly matching vectors. The formula for calculating cosine similarity is as follows:

$$\vec{u} \cdot \vec{v} = |\vec{u}| \cdot |\vec{v}| \cos \theta \quad (2)$$

$$\text{sim} = \cos \theta = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|} = \frac{\sum_{i=1}^n u_i \cdot v_i}{\sqrt{\sum_{i=1}^n u_i^2} \cdot \sqrt{\sum_{i=1}^n v_i^2}} \quad (3)$$

where \vec{u} and \vec{v} are the feature vectors of two text representations, and $|\vec{u}|$ and $|\vec{v}|$ are their respective magnitudes.

The directional evaluation of text-based applications through cosine similarity functions better than other methods because it addresses vector length normalization and enhances directional comparisons without considering vector size magnitude. The research uses cosine similarity to evaluate movie similarity based on Word2Vec dense vectors and CountVectorizer sparse vectors. After calculating similarity scores through computation tools, they get used for recommendation generation where higher scores represent greater relevance. This research analyzes the models' cosine similarity results which enables assessment of Skip-Gram's semantic connection abilities against CountVectorizer's frequency-dependent model.

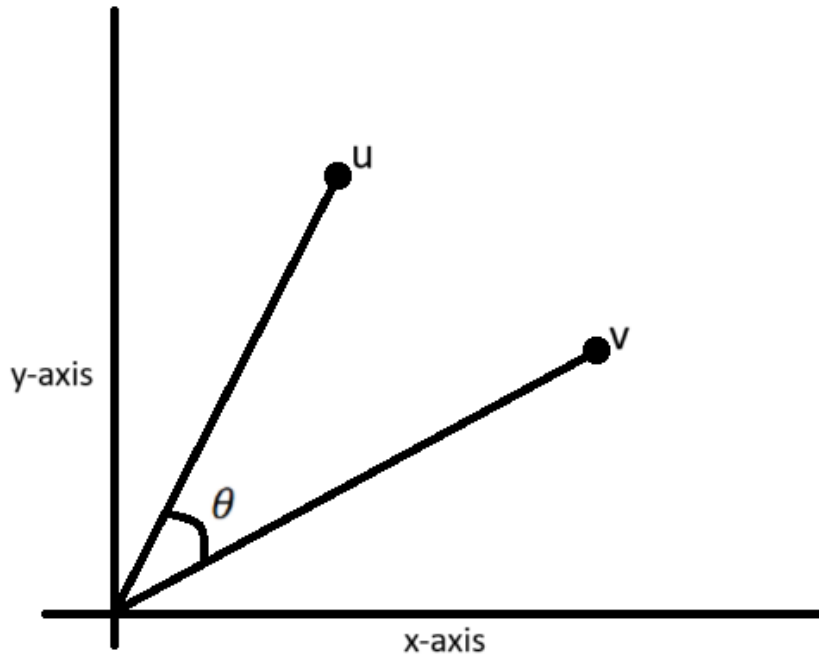


Fig. 7. Cosine Similarity

3.5. Hardware Requirements

A software application's basic hardware requirements vary depending on the kind of software being developed and the user's inclination towards programming tools like Python, Google Colab, Kaggle Notebook or Visual Studio Code. Applications with large object arrays might benefit from more RAM, but those that need faster processing for intricate activities or computations might need a CPU with greater performance.

Operating System	:	Windows 8/10/11
Processor	:	i5/i7
RAM	:	4/8 GB
Hard Disk	:	512 GB

3.6. Software Requirements

Practical demands and detailed description documents cover a wide range of topics, such as product perspective, features, operational framework, workspace, visual requirements, design limitations, and user manuals. These articles provide a thorough evaluation of the project, highlighting its advantages, disadvantages, and challenges in execution. By revealing issues and offering solutions, this information facilitates the growth process.

Operating System	-	Windows8/10/11
Programming Language	-	Python 3.8

3.7. Libraries

This chapter outlines in detail the key Python libraries and software tools utilized in the development of the content-based movie recommendation system presented in this thesis. The choice of Python stems from its high-level simplicity, extensive support for data science tasks, and an active open-source community that continually enhances its capabilities.

3.7.1. Python Software Ecosystem

Python is among the most widely adopted programming languages for machine learning, data analysis, and natural language processing due to its simplicity and versatility. It supports a variety of models ranging from traditional classifiers like

Support Vector Machines (SVMs) using Scikit-learn to deep learning models such as Convolutional Neural Networks (CNNs) implemented via TensorFlow and Keras.

While our project is focused on content-based recommendation using textual metadata, the broader capabilities of Python, including support for image and speech processing, underline its significance in comprehensive AI applications. The use of NLP tools like CountVectorizer and Word2Vec, integrated seamlessly into Python, supports our system's ability to extract and model semantic relationships from movie descriptions and keywords. Python's advantages extend further into rapid prototyping, platform independence, support for multiple programming paradigms (object-oriented, functional), and interoperability with other languages like C and C++.

➤ **Advantages**

- **Vast Library Ecosystem:** Python provides a massive collection of libraries for diverse functionalities—web scraping (BeautifulSoup), numerical computing (NumPy), data analysis (Pandas), machine learning (Scikit-learn), natural language processing (NLTK, Gensim), and deep learning (TensorFlow, PyTorch).
- **Simplicity and Productivity:** With its readable syntax and minimal boilerplate code, Python accelerates development. Tasks that require multiple lines in Java or C++ can often be completed in a few lines in Python.
- **Integration and Extensibility:** Python integrates easily with C/C++, Java, and .NET components, allowing performance bottlenecks to be addressed with lower-level optimizations.
- **Cross-Platform Portability:** Python scripts can be run across Windows, Linux, and MacOS with minimal changes.
- **Community and Open Source:** Its open-source nature means it is freely available and continually updated by a global community.
- **Object-Oriented and Functional Paradigms:** Python's support for both paradigms allows developers to choose the best approach for the problem at hand.

➤ **Disadvantages**

Although Python has many benefits, it's vital to take into account its drawbacks before deciding to use it for a project. The following are some drawbacks to consider:

- **Limitations of Speed:** In comparison to languages like C++ or Java, Python might be slower because of its interpreted nature. Applications that depend on performance could start to worry about this. However, Python's advantages exceed its drawbacks for the majority of general- purpose jobs.
- **Capabilities for browsing:** Python is less often used in client-side programming or in the creation of mobile apps and is mostly used on the server side. Although there are frameworks for running Python in browsers, such as Brython, their usage is restricted because of security issues. This limits the use of Python in some fields.
- **Limitations on Design:** Because Python has dynamic typing, variables can be assigned without explicitly defining their types. This flexibility can increase the efficiency of programmers, but if not used appropriately, it can also result in runtime issues. MyPy and other static type checking tools can help to lessen this problem.
- **Inadequate Connectivity to Databases:** The database access layers in Python are not as developed or commonly used as those in JDBC or ODBC. Large businesses with intricate database needs may want to take this into account as they may favour more well-established frameworks and technologies.

3.7.2. Project Modules/Labraries

➤ Scikit learn

Scikit-learn is important in this project since it supplies the major functions used in machine learning. Mainly, it is employed to perform CountVectorizer, a process that converts the movie data to numerical form for calculating similarity. The approach also has functions for computing cosine similarity between feature vectors, so movies can be compared by what's included in their content profiles. This framework uses an intuitive API and fast processes to make it very effective for building and launching filtering models.

➤ Matplotlib

Matplotlib is used to illustrate how different types of data are arranged and to show how similar or different vector values are. Although it does not produce recommendations, it is essential during the design and assessment of the system. You can use visuals to learn about the way movie vectors connect in the feature space and how well the clustering method groups identical content together. They make it easier to diagnose and adjust issues with how recommendations are made.

➤ **Pandas**

Pandas is commonly chosen when loading and processing data. It helps manage tables of data and manipulate movie information without difficulty. Performing activities such as processing CSV files, handling missing data, joining datasets and creating metadata strings is made simple with Pandas. The DataFrame structure plays a big role in making movie names and keywords well-suited for use in vectorization tools.

➤ **Numpy**

NumPy and Pandas work well together, since NumPy gives great functionalities for working with numbers. In the project, arrays and matrices are used, especially to find measures of similarity and organize vector data. Having the ability to quickly handle big arrays is very important for processing both CountVectorizer and Word2Vec data. Thanks to its efficiency and trustworthiness, NumPy is a main tool in the vector processing workflow.

➤ **Gensim**

NLP is very important for understanding and interpreting text data, mostly in content-based recommendation systems. Word2Vec and in particular the Skip-gram architecture are significant methods in NLP used to learn word representations. This project applies Word2Vec through Gensim, a Python library for NLP. Skip-gram uses a target word to predict related context words and thus finds semantic links between them when they appear in a window. In contrast to method relying on frequency data, Skip-gram can represent words along a continuous vector axis so that similar words are grouped closer in the space. Metadata such as movie titles and keywords are seen as input documents and the Skip-gram model's goal is to learn embeddings based on them. The results are used to form a vector for every movie, allowing the overall system to match up and suggest movies with comparable themes. Because Skip-gram is used, movies can be linked to recommend each other, even if their exact matches are sparse.

We used Gensim for this work to enable training of Word2Vec models using the written information of films. It supports the development of thick vector representations for words that include more than how often a term appears. Using Skip-gram and CBOW algorithms, Gensim gives users the ability to work with word contexts in different ways. The main advantage of Word2Vec embeddings is that they allow the system to choose movies with similar concepts.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1. Performance Evaluation Metric

For this chapter, we evaluate the effectiveness of our proposed Word2Vec-based movie recommendation system using widely used metrics from these two areas. Examples are Precision@k, Mean Reciprocal Rank (MRR), Mean Average Precision (MAP@k) and Normalized Discounted Cumulative Gain (NDCG@k) and 'k' in each case is the number of top recommendations reviewed. The implementation we choose uses 10 suggested movies to assess each metric.

Precision@10 calculates the percentage of the top 10 recommendations that are connected to the input film. This means you can see how accurate the user will be when making a decision. The MRR measures the position of the first useful result in the ranked list, telling us how efficiently the model can present the best choice. By calculating mean precision among all queries, MAP@10 judges which systems often rank higher those items that are the most useful. NDCG@10 shows how relevant each item is and also considers the position of all relevant items, assigning a bigger penalty to items ranked at the bottom to highlight the importance of how well the ranking is done.

We ensured top performance by adjusting the hyperparameters used for the Word2Vec Skip-gram model. The hyperparameters were set using a grid search approach on the number of dimensions for vectors (`vector_size`), the size of the training window (`window`), the minimum number of words needed (`min_count`) and the number of training epochs (`epochs`). The candidate values considered while tuning were gathered in Table 2:

Table 2. Parameter of Word2Vec Skip-Gram Model, Bold text is decided after hyperparameter-tuning.

Parameter	Values
<code>vector_size</code>	[10, 30, 50 , 100, 300]
<code>window</code>	[10, 20 , 30]
<code>min_count</code>	[1, 2 , 5]
<code>epochs</code>	[10 , 20, 30]

The hyperparameter configuration that delivered the best results was: `vector_size` = 50, `window` = 20, `min_count` = 2, and `epochs` = 10. This setup balanced vector quality and computational efficiency.

Following the hyperparameter tuning, we evaluated the model’s effectiveness using the aforementioned metrics. The results, presented in Table 2, demonstrate strong performance across all evaluation dimensions:

Table 3. Performance Metric after hyperparameter-tuning of the Word2Vec Model.

Metric	Value
Precision@10	0.782297852900868
MRR	0.8907395926240147
MAP@10	0.8400695485015148
NDCG@10	0.9042125372714982

These results indicate that the model performs well in recommending relevant movies, not only retrieving relevant items but also ranking them effectively. The high MRR and NDCG scores reflect the system’s ability to place relevant recommendations at the top of the list, thus enhancing user experience. The combination of strong MAP and Precision values suggests that the model maintains consistent performance across different user queries. This evaluation confirms the suitability of the Word2Vec-based content filtering approach for real-world movie recommendation tasks.

4.2. Result

Analysis reveals the effectiveness with which a system understands meaningful relationships between movies during this section. The recommendation model needs to imitate human thought by establishing connections between movies according to their themes alongside narrative elements and contextual context instead of basic word comparison. Streaming platforms today select Word2Vec deep learning models because they deliver contextually precise suggestions and ultimately give better recommendation services to users. In fig. 8, and 9 shows recommendation of “Thor”, and “The Avengers” movie respectively. CountVectorizer mainly functions through word frequency analysis and exact text matching with the dataset contents. The recommendation system based on CountVectorizer regurgitates movies that share identical terminology rather than movies with deep semantic meaning. Word2Vec goes beyond CountVectorizer because it recognizes word meaning by examining their patterns in context. Word2Vec SkipGram proves its capacity to match films that display matching themes together with comparable narratives and character profiles.

```
: getRecommendations_countVectorizer("Thor") # using CountVectorizer
```

```
Recommendations for 'Thor':
```

	Title	Similarity Score
0	Thor: The Dark World	0.800000
1	Ant-Man	0.734130
2	The Avengers	0.692820
3	Iron Man 2	0.688247
4	Avengers: Age of Ultron	0.640000
5	X2	0.603023
6	Captain America: The Winter Soldier	0.591864
7	X-Men	0.577350
8	Captain America: Civil War	0.548795
9	Captain America: The First Avenger	0.530791

```
: getRecommendations_Word2Vec("Thor") # using Word2Vec Skip-Gram
```

```
Recommendations for 'Thor':
```

	Title	Similarity Score
0	Thor: The Dark World	0.990217
1	The Avengers	0.980603
2	Ant-Man	0.979461
3	Avengers: Age of Ultron	0.978007
4	Iron Man 2	0.971432
5	X-Men	0.970298
6	X2	0.967356
7	X-Men: The Last Stand	0.963459
8	The Incredible Hulk	0.963069
9	Captain America: The Winter Soldier	0.962628

Fig. 8. Recommendations for "Thor" using CountVectorizer and Word2Vec SkipGram Models.

```
getRecommendations_countVectorizer("The Avengers") # using CountVectorizer
```

```
Recommendations for 'The Avengers':
```

	Title	Similarity Score
0	Avengers: Age of Ultron	0.769800
1	Ant-Man	0.750568
2	Iron Man 2	0.706417
3	Thor: The Dark World	0.692820
4	Thor	0.692820
5	X2	0.638285
6	Captain America: The Winter Soldier	0.636524
7	X-Men	0.611111
8	Deadpool	0.602464
9	The Incredible Hulk	0.601113

```
getRecommendations_Word2Vec("The Avengers") # using Word2Vec Skip-Gram
```

```
Recommendations for 'The Avengers':
```

	Title	Similarity Score
0	Avengers: Age of Ultron	0.981772
1	Thor	0.981151
2	Ant-Man	0.975361
3	Thor: The Dark World	0.973736
4	The Incredible Hulk	0.968656
5	Iron Man 2	0.967710
6	X-Men: Apocalypse	0.964204
7	Fantastic Four	0.960200
8	Captain America: The Winter Soldier	0.959281
9	X-Men	0.957144

Fig. 9. Recommendations for "The Avengers " using CountVectorizer and Word2Vec SkipGram Models.

CHAPTER 5

CONCLUSION

The research evaluated how movie recommendation systems change when using text-processing methods Word2Vec Skip-Gram and CountVectorizer. Simple word frequency in CountVectorizer cannot detect complex links between movies. Word2Vec Skip-Gram establishes contextual relations between words which produces recommendations that are more significant and natural to understand [15, 16]. The research indicates that deep learning algorithms that use Word2Vec prove superior to basic frequency-based techniques because they understand cinematic connections that extend further than general keyword overlap. Current real-world applications require the resolution of several obstacles including programming complexities and initial beginning hiccups and expansion limitations.

5.1. Future Work

The future improvement of recommendation accuracy can be achieved through integrating deep learning methods with collaborative filtering models according to [10]. New generation recommendation algorithms will enhance streaming service user experiences through better content recommendations for each viewer to find their preferred content.

REFERENCES

- [1] A. A. Joseph and A. M. Nair, "A Comparative Study of Collaborative Movie Recommendation System," 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 2022, pp. 1579-1583, doi: 10.1109/ICEARS53579.2022.9752015.
- [2] N. Taj, M. H. Varun and V. Navya, "Advanced Content-based Movie Recommendation System," 2024 5th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2024, pp. 1693-1698, doi: 10.1109/ICOSEC61587.2024.10722333.
- [3] N. P. Sable, A. Yenikar and P. Pandit, "Movie Recommendation System Using Cosine Similarity," 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), Pune, India, 2024, pp. 1-5, doi: 10.1109/I2CT61223.2024.10543873.
- [4] H. Khatter, N. Goel, N. Gupta and M. Gulati, "Movie Recommendation System using Cosine Similarity with Sentiment Analysis," 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2021, pp. 597-603, doi: 10.1109/ICIRCA51532.2021.9544794.
- [5] I. Naskar and N. P. Joseph, "Implementation of Movie Recommendation System Using Hybrid Filtering Methods and Sentiment Analysis of Movie Reviews," 2024 IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE), Bangalore, India, 2024, pp. 513-518, doi: 10.1109/ICWITE59797.2024.10502695.
- [6] Q. Pu and B. Hu, "Intelligent Movie Recommendation System Based on Hybrid Recommendation Algorithms," 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIIE), Ballari, India, 2023, pp. 1-5, doi: 10.1109/AIKIIE60097.2023.10389982.
- [7] S. S, M. S, S. R, S. M, S. M and S. S, "Certain Investigations on Cognitive based Movie Recommendation system using Pairwise Cosine Similarity," 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2023, pp. 2139-2143, doi: 10.1109/ICACCS57279.2023.10112790.
- [8] W. Zhao et al., "Leveraging Long and Short-Term Information in Content-Aware Movie Recommendation via Adversarial Training," in IEEE Transactions on Cybernetics, vol. 50, no. 11, pp. 4680-4693, Nov. 2020, doi: 10.1109/TCYB.2019.2896766.
- [9] Chollet, F., et al. (2015). Keras. GitHub. Retrieved from <https://github.com/fchollet/keras>.
- [10] S. Sahu, R. Kumar, M. S. Pathan, J. Shafi, Y. Kumar and M. F. Ijaz, "Movie Popularity and Target Audience Prediction Using the Content-Based Recommender System," in IEEE Access, vol. 10, pp. 42044-42060, 2022, doi: 10.1109/ACCESS.2022.3168161.

- [11] P. Mondal, P. Kapoor, S. Singh, S. Saha, J. P. Singh and A. K. Singh, "Genre Effect Toward Developing a Multi-Modal Movie Recommendation System in Indian Setting," in *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2517-2526, Feb. 2024, doi: 10.1109/TCE.2023.3324009.
- [12] S. Kumar, K. De and P. P. Roy, "Movie Recommendation System Using Sentiment Analysis From Microblogging Data," in *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 915-923, Aug. 2020, doi: 10.1109/TCSS.2020.2993585.
- [13] S. Kanwal, S. Nawaz, M. K. Malik and Z. Nawaz, "A Review of Text-Based Recommendation Systems," in *IEEE Access*, vol. 9, pp. 31638-31661, 2021, doi: 10.1109/ACCESS.2021.3059312.
- [14] M. Mngomezulu and R. Ajoodha, "A Content-Based Collaborative Filtering Movie Recommendation System using Keywords Extractions," 2022 International Conference on Engineering and Emerging Technologies (ICEET), Kuala Lumpur, Malaysia, 2022, pp. 1-6, doi: 10.1109/ICEET56468.2022.10007345.
- [15] Bird, Steven, Ewan Klein, and Edward Loper. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.", 2009.
- [16] Scikit-learn: Machine Learning in Python, Pedregosa et al., *JMLR* 12, pp. 2825-2830, 2011.
- [17] A. H. Izdiyar, N. D. Tsaniyah, F. Nurdini, B. R. Mufidah and N. A. Rakhmawati, "Building a Movie Recommendation System Using Neo4j Graph Database: A Case Study of Netflix Movie Dataset," 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETISIS), Manama, Bahrain, 2024, pp. 614-618, doi: 10.1109/ICETISIS61505.2024.10459699.
- [18] M. Manwal, D. Rawat, D. Rawat, K. C. Purohit and T. Choudhury, "Movie Recommendation System Using TF-IDF Vectorizer and Bag of Words," 2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 2023, pp. 163-168, doi: 10.1109/SMART59791.2023.10428182.



DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

PLAGIARISM VERIFICATION

Title of the Thesis MOVIE RECOMMENDATION SYSTEM USING WORD EMBEDDING

Total Pages 40 Name of the Scholar AMIT SINGH

Supervisor (s)

(1) DR. RAJESH KUMAR YADAV

(2) _____

(3) _____

Department COMPUTER SCIENCE AND ENGINEERING

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: TURNITIN Similarity Index: 7 %, Total Word Count: 9, 672

Date: 30/05/2025

Candidate's Signature

Signature of Supervisor(s)

Amit

Amit_Thesis.pdf



Delhi Technological University

Document Details

Submission ID

trn:oid:::27535:98218587

Submission Date

May 28, 2025, 10:56 PM GMT+5:30

Download Date

May 28, 2025, 11:02 PM GMT+5:30

File Name

Amit_Thesis.pdf

File Size

675.4 KB

40 Pages

9,672 Words

60,540 Characters





7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

-  **28** Not Cited or Quoted 7%
Matches with neither in-text citation nor quotation marks
-  **0** Missing Quotations 0%
Matches that are still very similar to source material
-  **0** Missing Citation 0%
Matches that have quotation marks, but no in-text citation
-  **0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 5%  Internet sources
- 3%  Publications
- 6%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 28 Not Cited or Quoted 7%**
Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 5% Internet sources
- 3% Publications
- 6% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	dtusimilarity on 2024-05-29	3%
2	Internet	www.coursehero.com	1%
3	Submitted works	CSU, San Jose State University on 2024-05-07	<1%
4	Internet	sallyfitzgibbonsfoundation.com	<1%
5	Internet	artificialintelligence-notes.blogspot.com	<1%
6	Internet	dspace.dtu.ac.in:8080	<1%
7	Internet	www.dominikkowald.info	<1%
8	Internet	exascale.info	<1%
9	Submitted works	Arab Open University on 2024-11-07	<1%
10	Submitted works	GLA University on 2015-05-02	<1%

11	Publication	Prabir Mondal, Pulkit Kapoor, Siddharth Singh, Sriparna Saha, Jyoti Prakash Singh...	<1%
12	Internet	ethesis.nitrkl.ac.in	<1%
13	Internet	www.compuserve.com	<1%
14	Publication	Amjad A. Alsuwailimi. "Arabic dialect identification in social media: A hybrid mod...	<1%
15	Publication	P. Pavan Kumar, S. Vairachilai, Sirisha Potluri, Sachi Nandan Mohanty. "Recomme...	<1%
16	Submitted works	University of Limerick on 2022-08-23	<1%
17	Internet	etd.uwc.ac.za	<1%
18	Internet	tel.archives-ouvertes.fr	<1%
19	Internet	thesai.org	<1%
20	Internet	umpir.ump.edu.my	<1%