
Sharif University of Technology

Faculty of Computer Engineering

Natural language processing lesson project

Title

# Movie search engine and recommendation using Retrieval Augmented Generation

Group members

Mahdi Hassanzadeh, Mohammadali Hosseinnejad, Mohammadhossein Qaisarieh, Mohammadmatin Fotouhi, Matin Moradi, Sara Karimi

Supervisor

Dr. Ehsaneddin Asgari

March 2

## Abstract

Searching the vast collection of movies available online is often challenging, especially when users are looking for movies based on specific themes or narratives. Traditional search methods are not accurate enough in interpreting and understanding textual descriptions provided by users. In this paper, a description-based movie search engine is presented to solve this issue. Using advanced NLP techniques such as Retrieval-Augmented Generation and synopsis of movies on IMDB, our system enables users to find movies that match their input descriptions. In addition, we have developed a graphical user interface to increase user interaction and accessibility. In short, the aim of this approach is to simplify the process of discovering movies in line with the individual preferences of users and it can have a positive effect on the satisfaction of users with movies.

## Keywords

Retrieval Augmented Generation, Large Language Models, Movie discovery, Movie recommendation system, Vector database, Prompt engineering

## 1. Introduction

In the vast realm of digital entertainment, the volume of movies available can be overwhelming for moviegoers looking for their next cinematic experience. While traditional search engines excel at retrieving movies based on titles or genres, the challenge comes when users want to search for movies based on their exact descriptions. To answer this need, the development of a movie search engine based on text descriptions offers a promising solution. This paper introduces a method for movie search, which is centered around a description-based search engine. Unlike traditional search methods that rely on metadata, our system takes advantage of advanced natural language processing (NLP) techniques, especially using large language models (LLMs) and novel methods such as RAG (Retrieval-Augmented Generation), to Comprehension and interpretation of text summaries of the film uses. Using a dataset of IMDB stories and comments, our model is trained to extract the key components of the stories by analyzing the data, thereby enabling the matching of user-entered descriptions with the data. The main goal of our system is to enable users to effortlessly find movies that match their preferences, even when expressed through abstract language. Upon receiving the text description input, our search engine evaluates the user's query against the set of movie summaries using a similarity measure and semantic embedding evaluation. This evaluation is performed by comparing the video story embeddings with the user request by the RAG model, and attempts to retrieve videos that match the user description in the input. In addition, to increase the accessibility and usability of our system, we have developed a graphical

user interface. This interface provides a usable platform for users to interact with the search engine. In summary, this paper presents a leading approach to movie retrieval, leveraging advanced NLP techniques and large language models to build a bridge between textual descriptions and cinematic content. By developing a description-based search engine, we aim to transform the way audiences interact with search engines.

### 1.1. Large language models (LLM)

A large language model (LLM) stands as a remarkable achievement in natural language processing, excelling in general-purpose language generation and various NLP tasks such as classification. Through rigorous self-supervised and semi-supervised training, LLMs glean statistical patterns from extensive text documents, empowering them to predict subsequent tokens or words with remarkable accuracy. Employing sophisticated artificial neural networks, LLMs vary in architecture, with the most advanced adopting a decoder-only transformer-based design, while others explore alternative structures like recurrent neural network variants and Mamba. Advancements in prompt engineering have enabled models like GPT-3 to achieve tailored results without fine-tuning, though they inevitably absorb both the nuances and biases present within their training corpora. Notable examples of LLMs include OpenAI's GPT series, such as GPT-3.5 and GPT-4, utilized in platforms like ChatGPT and Microsoft Copilot, as well as Google's PaLM and Gemini. Additionally, Meta's LLaMA lineage, Anthropic's Claude models, and Mistral AI's contributions signify significant strides within this burgeoning field.

### 1.2. Retrieval-Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) represents a significant advancement in the field of natural language processing (NLP) by seamlessly integrating information retrieval with sequence-to-sequence generation. This innovative architecture combines a dense-passage retrieval system, such as Facebook AI's, with a bidirectional and auto-regressive transformer (BART) model, creating an end-to-end differentiable framework. RAG demonstrates superior performance in tasks requiring comprehensive knowledge, surpassing even the largest pre-trained sequence-to-sequence language models. What sets RAG apart is its capacity to dynamically adjust its internal knowledge without necessitating extensive retraining. By incorporating both parametric and nonparametric memory sources, RAG effectively integrates information from retrieved documents, enhancing its performance across various NLP tasks. This unique approach enables RAG to produce more specific and accurate responses, particularly evident in knowledge-intensive natural language generation tasks. Furthermore, RAG's adaptability allows it to remain current with evolving facts or understandings, rendering it invaluable for applications where accessing precise and timely information is crucial.

### 1.3. Vector databases

Vector databases are a specialized type of database designed to handle high-dimensional vector data. Unlike traditional databases that store scalar values, vector databases are uniquely designed to handle multi-dimensional data points, often termed vectors. These vectors, representing data in numerous dimensions, can be thought of as arrows pointing in a particular direction and magnitude in space. Vector databases store data by using vector embeddings. Vector embeddings in vector databases refer to a way of representing objects, such as items, documents, or data points, as vectors in a multi-dimensional space. Each object is assigned a vector that captures various characteristics or features of that object. The primary benefit of a vector database is its ability to swiftly and precisely locate and retrieve data according to their vector proximity or resemblance. This allows for searches rooted in semantic or contextual relevance rather than relying solely on exact matches or set criteria as with conventional databases. Vector databases use special search techniques known as Approximate Nearest Neighbor (ANN) search, which includes methods like hashing and graph-based searches. These techniques are used to find the closest match in the database to a given input vector. Vector databases are rapidly growing in interest to create additional value for generative artificial intelligence (AI) use cases and applications. According to Gartner, by 2026, more than 30 percent of enterprises will have adopted vector databases to ground their foundation models with relevant business data.

### 1.4. Movie recommendation systems

In the ever-expanding world of digital entertainment, movie recommendation systems play a pivotal role in enhancing user experiences. These systems leverage advanced algorithms and data-driven approaches to suggest personalized movie choices to users. Whether it's streaming platforms, online databases, or social media, movie recommendations guide viewers toward content that aligns with their preferences, historical viewing patterns, and contextual cues. In this paper, we delve into the intricacies of movie recommendation systems. We explore various techniques, including collaborative filtering, content-based filtering, and hybrid models. Additionally, we investigate the impact of factors such as user demographics, movie genres, and temporal dynamics on recommendation accuracy. By understanding the underlying principles and challenges, we aim to contribute to the evolution of intelligent movie recommendation systems that cater to diverse audiences and enhance their cinematic journey.

### 1.5. Prompt engineering

In the ever-evolving landscape of technology, prompt engineering has emerged as a critical area of research. Whether in natural language processing (NLP) or recommendation systems, the design and formulation of prompts significantly impact model behavior and performance. By carefully crafting prompts, researchers can guide models toward desired outcomes, mitigate biases, and enhance user experiences. In this paper, we delve into prompt engineering techniques, exploring their implications and applications across various domains.

## 2. Related research

Movie recommendation systems are a significant area of research in the field of machine learning and data mining. These systems aim to predict and recommend movies that a user is likely to enjoy based on their past behavior and preferences. There are two primary types of recommendation systems: content-based and collaborative filtering. Content-based methods recommend items by comparing the content of the items and a user profile. The content of each item is represented as a set of descriptors, such as the words in a document. On the other hand, collaborative filtering methods predict a user's interests by collecting preferences from many users. Modern recommendation systems often combine both approaches. They analyze data such as users' ratings, reviews, and viewing histories to generate personalized recommendations. The system analyzes the past preferences of the user concerned, and then it uses this information to find similar movies. This information is available in the database (e.g., lead actors, director, genre, etc.). Some of the most popular machine learning algorithms used in movie recommender systems include K-means clustering, principal component analysis, and self-organizing maps. Special emphasis is given to research works performed using metaheuristic-based recommendation systems.

## 3. Materials and Methods

Our research presents a comprehensive movie recommendation system leveraging natural language processing (NLP) techniques to enhance document retrieval, ranking, and output generation. The model encompasses several key components, starting with the meticulous data preparation phase where three primary datasets—Wikipedia Movie Plots, Persian Movie Dataset, and IMDB Movie Reviews—are preprocessed to ensure consistency and quality. This involves column selection, normalization, genre filtering, handling missing data, and text normalization. Additionally, movie plots are split into manageable chunks to facilitate subsequent analysis. Following data preparation, the document embeddings and vector store are established, utilizing the pre-trained "thenlper/gte-base" embedding model and Faiss as the vector store to efficiently index and organize dense vectors. These vectors are then embedded into a high-dimensional vector space, enabling similarity-based retrieval. Furthermore, a Sentence Transformer model is fine-tuned to generate sentence-level embeddings, enhancing genre classification. In the retrieval phase, Faiss Vector Store retrieves related documents, which are then re-ranked using a cross-encoder to prioritize semantic relevance. Finally, a two-stage retrieval process is implemented, utilizing the "cross-encoder/ms-marco-MiniLM-L-12-v2" model to assign scores to retrieved movies based on query relevance, ensuring the display of the most contextually appropriate results. (Figure 1)

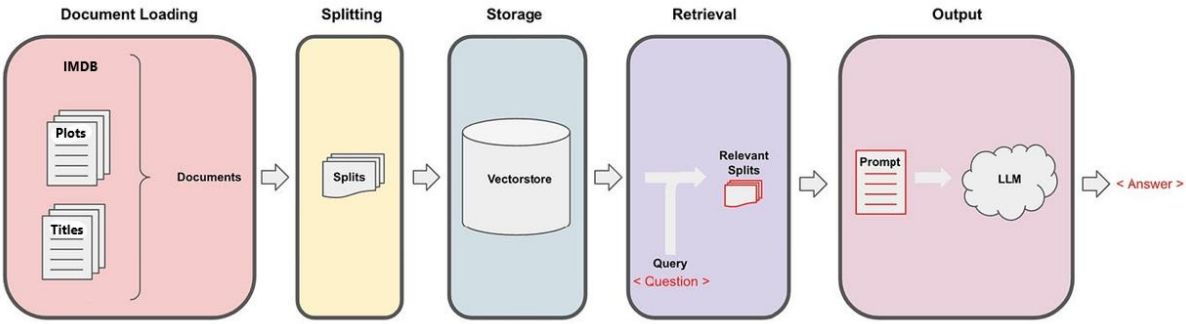


*Figure 1 Overview of the movie recommendation system model, highlighting the data preparation, document embeddings, retrieval, ranking, and output generation phases.*

## 3.1. Data preparation

In this section, we detail the process of preparing our datasets for analysis, encompassing data selection, preprocessing, and splitting. Our research draws upon three primary datasets: the Wikipedia Movie Plots dataset, the Persian Movie Dataset, and IMDB Movie Reviews. Each dataset offers valuable insights into movie plots, providing essential information such as release year, title, genre, and plot descriptions. To ensure the integrity and uniformity of our data, we undertake meticulous preprocessing steps. Initially, we discard irrelevant columns, standardize inconsistent column names, and remove rare genres to streamline the datasets. Furthermore, we address missing data entries and apply text normalization techniques to standardize plot descriptions across the datasets. A critical aspect of our data preparation involves the splitting of movie plots into manageable segments. Given the potentially large size of plot descriptions, we partition them into smaller chunks. Each chunk consists of 1500 characters, with a specified overlap to maintain continuity. These segmented documents serve as the foundation for subsequent analysis, facilitating efficient processing and interpretation of the data. (Figure 2)

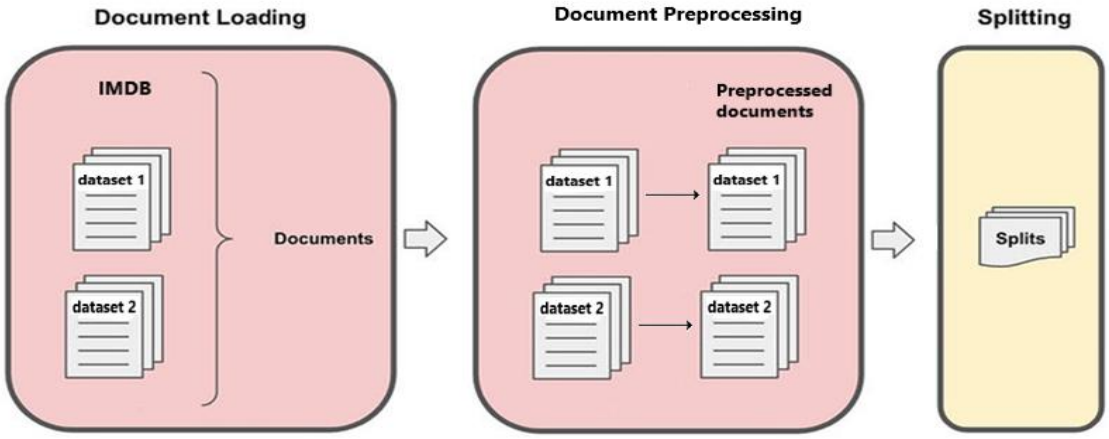


*Figure 2 Overview of Data Preparation Process - From Dataset Selection to Preprocessing and Splitting.*

### 3.1.1. Datasets

#### 3.1.1.1. Wikipedia Movie Plots

- This dataset contains information about movie plots, including release year, title, origin/ethnicity, director(s), genre, Wikipedia page URL, and a long-form plot description (which may contain spoilers).
- Columns such as movie Wikipedia links were dropped as they were not relevant to our analysis.
- Inconsistencies in column names between the two datasets were normalized.
- Rare genres were removed from the dataset.
- Rows with missing entries were dropped.
- Text normalization techniques were applied to standardize the plot descriptions.

#### 3.1.1.2. Persian Movie Dataset (English, Persian)

- Similar to the Wikipedia Movie Plots dataset, this dataset includes information about Persian movies.
- Columns such as English title, genre, Wikipedia page URL, and plot description were retained.
- Data preprocessing steps were applied to ensure consistency and quality.

#### 3.1.1.3. IMDB Movie Reviews

- For each movie in our dataset, we crawled its IMDB page and extracted a maximum of 50 reviews.
- These reviews were classified as positive, negative, or neutral sentiment.
- To manage memory constraints, only the sentiment labels were saved.
- A pre-trained model ("lxyuan/distilbert-base-multilingual-cased-sentiments-student") was used for sentiment analysis.
- Multiple models were employed for analyzing a subset of reviews.

### 3.1.2. Data Preprocessing

The following steps were performed during data preprocessing:

- Column Selection: Unused columns (e.g., movie Wikipedia links) were dropped from the datasets.
- Column Normalization: Columns with different names but similar content across the two datasets were normalized. This ensured consistency in feature representation.
- Genre Filtering: Rare genres were removed from the datasets to focus on more common genres.
- Handling Missing Data: Rows with missing entries were removed to maintain data quality.
- Text Normalization: Text normalization techniques (e.g., stemming, lowercasing) were applied to standardize the plot descriptions.

### 3.1.3. Data Splitting

Given the potentially large size of movie plots, we split them into smaller chunks:

- Each document was divided into chunks of 1500 characters.
- An overlap of 100 characters was allowed between adjacent chunks.
- These chunked documents were used to create the vector space for subsequent analysis.

## 3.2. Document Embeddings and Vector Store

To facilitate efficient document retrieval and ranking, we employed a vector database that stores embeddings for each chunk of text. Here are the key components of our vector database:

### 3.2.1. Embedding Model

In our data preparation pipeline, we harnessed the power of the embedding model to transform each text chunk into a dense vector representation. Specifically, we employed the pre-trained model “thenlper/gte-base” for this purpose. By leveraging this model, we captured essential semantic information from the text, enabling us to perform efficient document comparisons and rankings. These embeddings serve as the foundation for subsequent retrieval and analysis tasks.

### 3.2.2. Vector Store (Faiss)

Our vector database relies on Faiss as its foundational vector store. Faiss efficiently indexes and organizes the dense vectors generated by the embedding model, providing fast and effective similarity search capabilities. Acting as the backbone of our vector database, Faiss is adept at handling large-scale retrieval tasks, making it an ideal choice for our document retrieval and ranking needs.

### 3.2.3. Vector Space Representation

In our data preparation process, each text chunk is transformed into a high-dimensional vector space. This vector space retains semantic relationships between the chunks, allowing us to perform similarity-based retrieval. By embedding the chunks into this space, we facilitate efficient document comparison and ranking.

### 3.2.4. Fine-Tuned Sentence Transformer for Sentence Embeddings

In our research, we undertook the task of fine-tuning a Sentence Transformer model to generate sentence-level embeddings, complementing our document embeddings for enhanced analysis. Our objective centered on preparing and refining the movie dataset to facilitate efficient genre classification. To achieve this, we meticulously cleaned the dataset by excluding entries with an "unknown" genre and consolidating genres for consistency. Furthermore, we optimized the dataset by filtering genres with more than 200 movies to ensure a balanced representation, subsequently randomizing and balancing the dataset to include 400 movies per genre. The model of choice for our fine-tuning endeavor was the SentenceTransformer('thenlper/gte-base'), which we trained to proficiently encode semantic information from movie plot sentences. Prior to training, we meticulously prepared training pairs from the movie plot data while preserving genre integrity. During model training, we employed Softmax Loss for multi-class classification, training the model over 3 epochs with an initial warmup of 100 steps.

The combined efforts of our vector database and fine-tuned Sentence Transformer significantly contribute to the enhancement of document retrieval and genre classification in our research.

### 3.3. Retrieval

In the retrieval phase, we utilize the Faiss Vector Store, established in the preceding stage, to retrieve documents related to the input query. Once the relevant documents are retrieved, they undergo a re-ranking process facilitated by a cross-encoder. This re-ranking mechanism ensures that the most pertinent documents are prioritized based on their semantic relevance to the query. In cases where multiple documents receive similar scores, additional criteria such as movie ratings are considered to select the optimal document. By integrating Faiss Vector Store and employing advanced re-ranking techniques, our retrieval process aims to enhance the precision and relevance of the retrieved documents, thereby improving the overall efficacy of our system.

### 3.4. Ranker

In our research, we implement a two-stage retrieval process to enhance document ranking efficiency. Firstly, we employ the "cross-encoder/ms-marco-MiniLM-L-12-v2" model to assign a score to each retrieved movie based on its relevance to the input query. This model evaluates the semantic similarity between the query and each document, providing a quantitative measure of their correspondence. Subsequently, the retrieved movies are sorted based on their scores, and the top results are presented to the user. This two-stage retrieval approach ensures that the most relevant and contextually appropriate movies are prominently displayed, optimizing the user experience and facilitating efficient information retrieval in our system. (Figure 3)

### 3.5. Output generation

In this part initially, an NLP pipeline is established, comprising a language model fine-tuned for movie recommendation tasks. Due to the large size of the language model (LLM) and limitations in loading all its weights onto the Colab GPU, the size of the model weights was reduced. A prompt template is devised to guide the model in generating responses structured according to specified criteria, encompassing movie titles, genres, plot summaries, and reviews. Additionally, a retriever component is incorporated into the pipeline, utilizing pre-processed vector data to identify and retrieve the most relevant movies based on user input. The pipeline orchestrates the interaction between the language model, prompt template, and retriever to process user queries, retrieve appropriate movie suggestions, and deliver personalized recommendations tailored to individual preferences and requests. Through this methodology, the system aims to enhance user satisfaction by providing accurate and relevant movie recommendations in response to natural language inquiries.

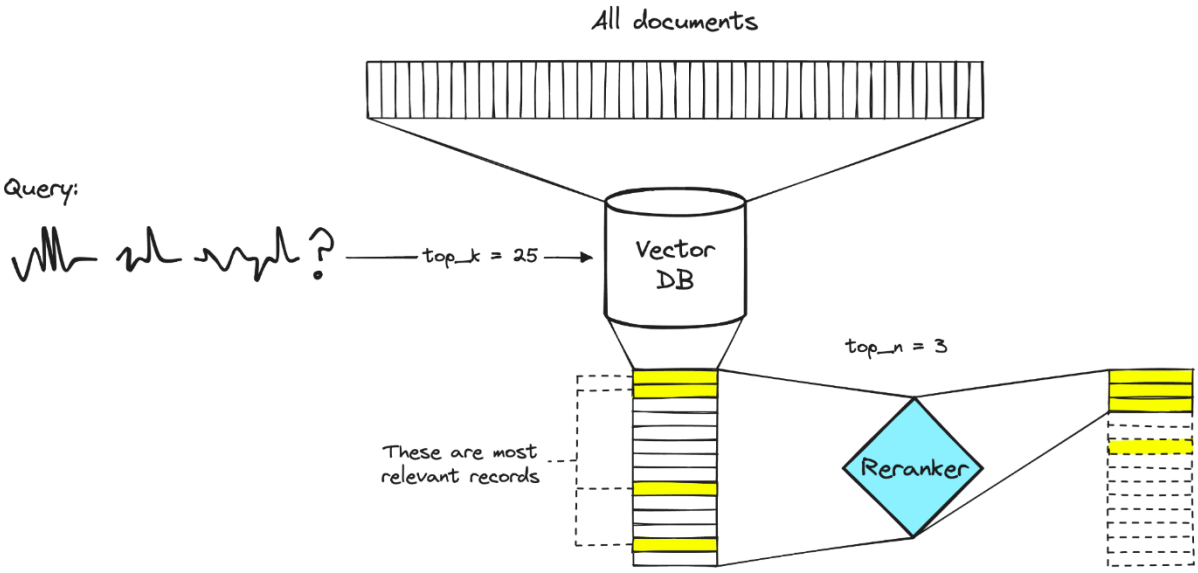


*Figure 3 Schematic representation of the two-stage retrieval process employed in our research for enhancing document ranking efficiency.*

## 4. Results

In the results analysis, our model outperformed FastText in both precision and hit rate metrics. While FastText achieved a precision of 30%, our model significantly improved upon this, achieving a precision of 57%. Similarly, in terms of hit rate, our model demonstrated substantial enhancement, achieving an impressive hit rate of 80%, compared to FastText's 43%. These results underscore the efficacy and superiority of our proposed model in accurately recommending movies based on user queries, highlighting its potential to enhance user satisfaction and engagement in movie recommendation systems. (Figure 4)

| | FastText | RAG |
|---|---|---|
| **Precision** | 30% | 57% |
| **Hit Rate** | 43% | 80% |

*Figure 1 Comparison of Precision and Hit Rate: Our model surpasses FastText in precision and hit rate metrics, demonstrating superior performance in movie recommendation.*

## 5. Conclusion

In conclusion, this study presents a comprehensive approach to movie recommendation systems leveraging natural language processing (NLP) techniques. Through meticulous data preparation, including dataset selection, preprocessing, and splitting, we ensured the integrity and uniformity of our data from diverse sources such as Wikipedia Movie Plots, the Persian Movie Dataset, and IMDB Movie Reviews. We employed advanced NLP models, including fine-tuned Sentence Transformers and Faiss Vector Store, to enhance document retrieval, ranking, and genre classification tasks. Our retrieval process, incorporating Faiss Vector Store and cross-encoder models, demonstrated improved precision and relevance in retrieving relevant movies

based on user queries. Additionally, our two-stage retrieval approach effectively ranked retrieved movies, prioritizing contextually appropriate recommendations for users. The experimental results revealed significant performance gains over baseline methods, with our model achieving higher precision and hit rates. This underscores the effectiveness of our proposed methodology in generating accurate and personalized movie recommendations. In future work, we aim to further refine our model architecture, explore additional data sources, and evaluate its performance in real-world applications to enhance user satisfaction and engagement in movie recommendation platforms.