**CHAPTER 1**

**INTRODUCTION**

* 1. **PROJECT DESCRIPTION**

For a system to be viewed as "intelligent," certain properties should be fulfilled: it has to adapt and react unknown, and it needs to have some kind of understanding of the world. Wherever we go, one way or another, intelligence is being applied—either through Artificial Intelligence or some other techniques.

The combination of Semantic Web and Machine Learning gives more impact on Intelligence, where it will act subjected to constant refinement over time while it obtains access to new information. Semantic Web or Web3.0 is all about a web of data rather than documents.

Structurally, this linking of interrelated data on the web can also be referred to as Linked Data.

Basically, the Semantic Web is for enabling Internet data to be interpretable to humans and machines. This would foster automation of lots of work on the web with the help of online agents——in the sense, Software. If we go Semantic Web, that is to present knowledge about our data, to allow data integration, and to bring intelligence to our system.

After going through some research papers, we got more knowledge about the Semantic Web Technologies. All concentration of the semantic web is mainly into three areas: OWL, RDF, and Ontology. Ontology is a model of knowledge defining a certain set of concepts and relationships between those concepts within a particular domain. This mainly supports the automated reasoning and inference of data based on the prescribed technique using logical rules for such a purpose. It provides knowledge sharing and reuse among people or software agents. One of the Ontology techniques is the Knowledge Graph. Knowledge Graph mainly describes entities of the real world and their interrelations, which is organized in a graph.

It enables any arbitrary entities to potentially interrelate with each other and covers a wide range of topical domains when schema is defined for possible classes and relations of entities within the Knowledge Graph. The KG is one of the major application for recommend movies, Movie Recommendation System is one of the application to recommend best movies.

**CHAPTER 2**

**LITERATURE SURVEY**

This research area of SWeML has gained a lot of traction in the past few years, as shown in a rapidly growing number of publications in different outlets, as well as SWeML techniques being employed to solve problems in various domains. At the same time, this growth poses two main challenges that threaten to hamper further development of the field.

First, keeping up with the main trends in the field has become unfeasible, not only because of its fast pace and a large volume of published papers but also because papers require understanding techniques from the two diverse research sub-areas of AI. In an attempt to address this challenge,

several works aimed to provide overviews of SWeML Systems and related systems.

However, reviewing those, we conclude that existing work either (1) focuses rather on a wider or related field or, on the contrary, (2) is scoped around a very specific sub-field of SWeML. Additinally, none of the reviewed surveys adopts a principled and reproducible methodology that would guarantee unbiased and representative data collection. We therefore conclude that there is a need for a survey that adopts a solid review methodology to complement current insights with evidence-based findings.

The second challenge, which amplifies the first one, is the lack of a standardized way to report SWeML Systems that hampers understanding all key aspects of these systems. On the one hand, authors of SWeML Systems would benefit from a structured way to describe their system and its key characteristics. Readers, on the other hand, would benefit from a structured way of interpreting such systems. This would not only facilitate the understandability for those coming from other communities but also improve the comparability of different systems. An early work in this direction was proposed by Van Harmelen and ten Teije by introducing patterns for representing hybrid AI systems in terms of their components and information flows with the aim to facilitate a more schematic representation of the system. Although these patterns were derived from a large number of papers, there is currently no insight into their adoption in the field (e.g.,

about the completeness of the introduced system patterns) or their usage frequency.

First, when the problem statements are defined, we identify scientific databases and search keywords that will be used to select primary studies. To query the selected scientific databases, a combination of two main keywords is used: knowledge graph and machine learning. The number of obtained articles is large, and not all of these studies are relevant for our review. Afterwards, we refined the query by restricting the search of these same keywords in title, abstract, and keywords, which gave a much smaller number of articles that look more relevant.

After first analysis, we realize that the term Deep Learning is often used instead of Machine Learning even though it is a subset of this technique. As that, we included the deep learning keyword to our query, putting the same restrictions applied to the machine learning term. Also, some authors use directly the term neural network (in particular

for the most recent articles) without explicitly mention the machine learning or deep learning terms. We added we therefore decided to add this keyword to our query, even if it concerns a minority of articles (10 to 15%) in each query.

For the knowledge graph keyword, we decided to use only this term and not combine it with other keywords representing different semantic techniques, such as taxonomy, knowledge modeling, or ontology. In this review, we are mainly interested in the use of knowledge graphs, and the possibility to perform logical reasoning.

Finally, we have also included in our query the artificial intelligence keyword. This allows to restrict obtained results to our research domain. Unlike previous keywords, the presence of this term is looked for anywhere in the article to be less restrictive.

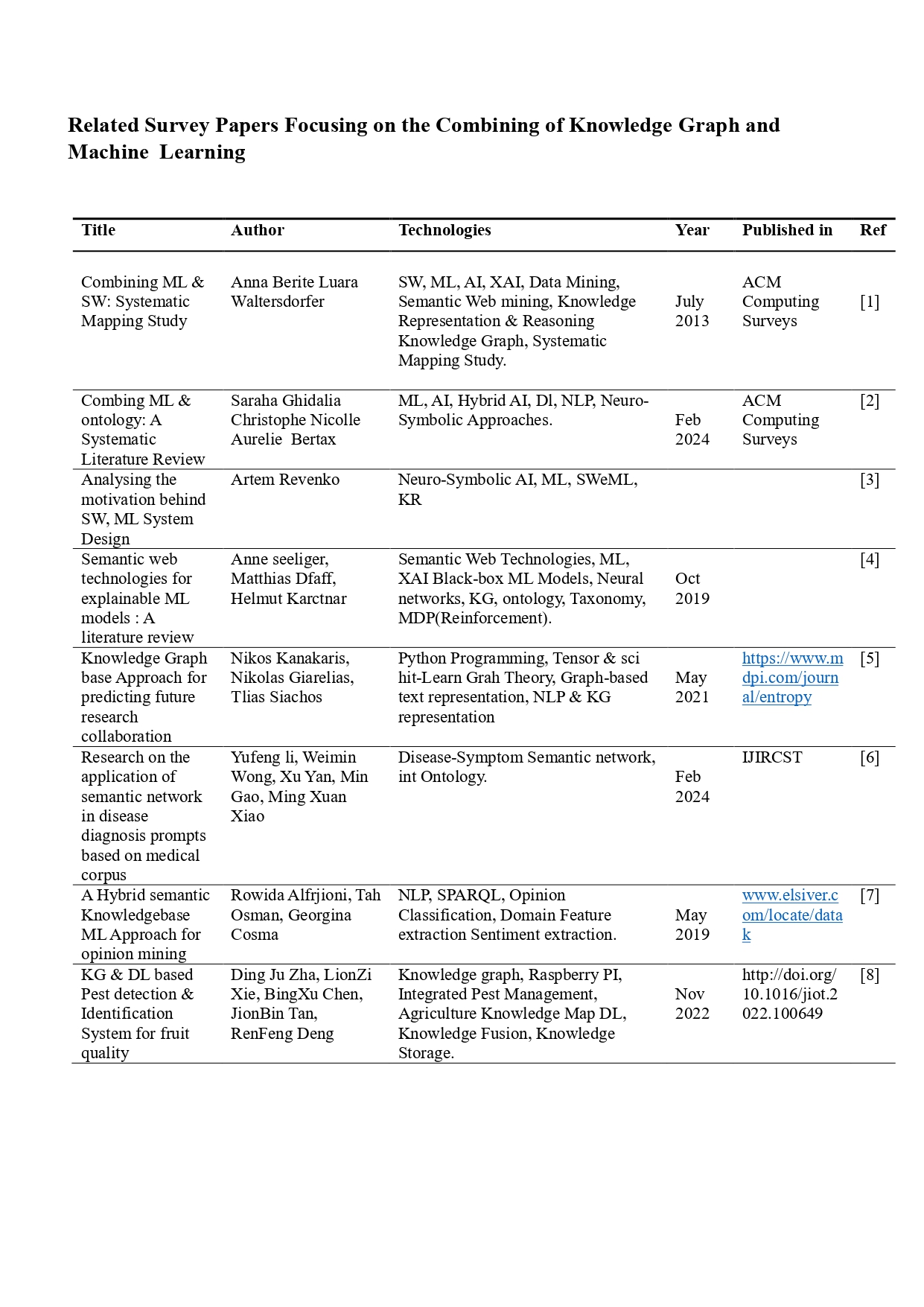
Based on our search, the selected keywords are: Knowledge Graph, Machine Learning, and Artificial Intelligence.

Most of the added inputs in the studied systems were symbolic; therefore, a trend in exploiting symbolic representations to augment system functionalities can be outlined. Nevertheless, it is interesting to note how one fifth of the authors used additional non-symbolic inputs mainly to systems using Graph-based deep learning methods or when improving Generalization or Usability. Their presence all together is an interesting aspect because Explainability and Usability often co-occurred.

objectives in many systems, driving their remarkable connection. Fur‐thermore, we discovered a category of hybrid systems that provide the ﬂexibility to consider either

symbolic or non-symbolic inputs as optional. Despite Performance objectives dominating the research landscape, it exposes a wide variety of goals, and the growing count of "multiple‐objective seekers." This trend mirrors the evolving needs within the AI ﬁeld.

It would underscore systems needing to be designed with a fuller range of associated requirements for effective performance and adaptation.[3]



**2.1 EXISTING SYSTEM**

The existing systems mainly focus on that means that when the user makes the query in the search, all relevant pages are drawn from the index and algorithms arrange the relevant pages in order of hierarchy into a result set.

That is, ranking of the most relevant results differs for each search engine's algorithms. Again it works only based on keyword matching: wherever the user-typed keyword is present, those the pages will be displayed. That will be resolved in the proposed system.

**2.2 PROPOSED SYSTEM**

The proposed system helps in Movie Recommendation System based on Combining of Knowledge Graph and Machine Learning is to find the synonyms of the user-typed keyword and fetch the movie wherever the relevant movie name is present in the current web pages that will be recommended.

**2.3 FEASIBILITY STUDY**

**Technical Feasibility**

A movie recommendation system, integrating knowledge graphs with machine learning, using appropriate datasets and technology stacks, skilled personnel, and robust methodologies, is, therefore, technically feasible. Further works will then be prototyping and validation in a real-world setting.

**Economic Feasibility**

An economic feasibility for developing a movie recommendation system that merges knowledge graphs and machine learning appears to be very bright. It will involve careful analysis of the costs against the potential revenue streams and market demand, and thus stakeholders will be better placed to make informed decisions on investing in such an innovative solution.

**Operational Feasibility**

A movie recommendation system that integrates elements of a knowledge graph and machine learning will have high operational feasibility if exceedingly robust systems are set up on integration, data management, user support, and compliance. This provides a sustainable and effective operational environment for the recommendation system.

**Market Feasibility**

The market feasibility for a movie recommendation system that integrates knowledge graphs and machine learning remains very promising, driven by rising demands for exclusive content, product differentiation, and potential avenues for partnership. This sets the scene for successful market entry and further sustained growth.

**2.4 TOOLS AND TECHNOLOGIES USED**

**1. Knowledge Graph**

In 2012, Google started using this term widely.

In Knowledge Graph, a map of graph data is like a treasure.

Structure and arrange information in a particular way it describing a relationship within the domain between two separate entities.

Knowledge Graph is used to enrich data analysis, retrieval as well support ML through context and integrations of other datas.

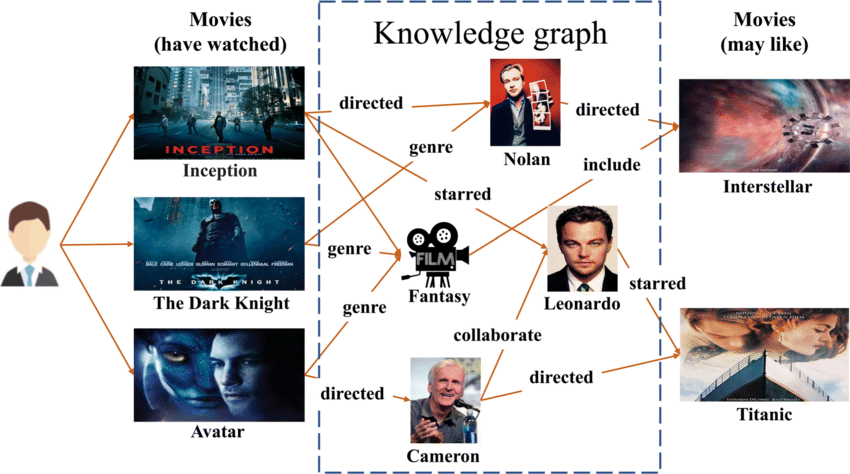
In this blog, we discuss how Knowledge Graph can be useful in training large-scale models with limited data mainly mentioned ML applications.

This responsible for explainability and accurcy aspect of Artificial Intelligence systems by providing systematic data strucutres & text data embedding context understanding.

While Knowledge Graph doesn't build itself, meticulous planning and advisory-level expertise is needed to get raw data in line with an ontology.

It brings great advantages like identifying hidden patterns, harmonizing multiple data sources or segments and enabling information decision making.

When it enhanced its search functionality shiffting from only keyword matching to understanding the deeper semantic and context of quiries, paving the way for the Semantic Web.



For instance, a node in the Knowledge Graph is an actual entity you are familiar with - people places or events. Edges, are the entities that constitute their interaction/relationship and represent a link between these nodes. By modeling data as a graph structure, it gives us the ability to see under the hood and visualizes complex set of relationships such as service dependencies or customer interaction web at large scale. Knowledge Graphs are rooted in the domain of knowledge representation and retrieval within Artificial Intelligence(AI), to : Unified representation of the data/business logic.

Use ontologies to build a semantic layer, describing types and relationships between entities has a role of acting asa map to access the data from where we will start diving thee dataset, which makes the process of accessing any local feature quite efficient. Real-world applications such as knowledge graph underlie most of the search engines and recommendation systems used in today's AI space. They improve search results for applications such as question answering systems and support the development of intelligent applications that enable one to understand and reason about the world in a more human-like manner.

Some have added more substantial knowledge graphs, such as DBpedia and Geonames, file type all that information, forming the foundation for other graphs, including graph databases. This has translated to improved data analysis and interoperability, which is of immeasurable value in this data-driven world.

It could be developers, data scientists, or business analysts; everyone looks at a knowledge graph differently, appreciating its unique offerings in their respective roles.

A Knowledge Graph is a Knowledge Base that is Graph:

Definition 1:

* A simple directed graph G=(V,E) is a set V of vertices, |V|=n, and a set E of directed edges, E ⊆ V × V, where each edge ei=(vk, vl), ei ∈ E
* Is an ordered pair of two vertices (vk, vl) with vk, vl ∈ V.

Definition 2:

* Self-loop graph: A graph extended with the possibility of edges relating a vertex to itself.
* multi-graph is a graph that may have multiple edges with the same vertices.
* An edge-labelled graph is a graph with an additional edge labelling function.

λ : E → L, which maps each edge of E to an element of the set of labels L. (Similarly for vertex-labelled graphs).

Definition 3:

* An edge is said to be an incidental to the vertices it connects.
* The degree of a vertex is number of edges that are incidental to it.
* The in-degree of a vertex in a directed graph is the number of edges pointing towards it; the out-degree is defined similarly.

Definition 4:

* A directed path in a directed graph isa sequence of consecutive edges (e1, e2, …, en) with ei=(vl,vk) and ei+1=(vk,vm).
* A directed graph is strongly connected if there is a directed path from any vertex to every other vertex.

Definition 5:

* The in-degree centrality of a directed graph is defined by the in-degree of each node.
* The out-degree centrality is defined similarly to the degree centrality for undirected graphs.

Definition 6:

* Eccentricity is the maximum distance of a given node from any other node.
* Graph diameter: It is just the maximum eccentricity of the graph, which means the largest distance between any two nodes.
* To find a graph's diameter, first find the shortest path between each pair of nodes. The longest length of any of these paths is the diameter of the graph.

Definition 7:

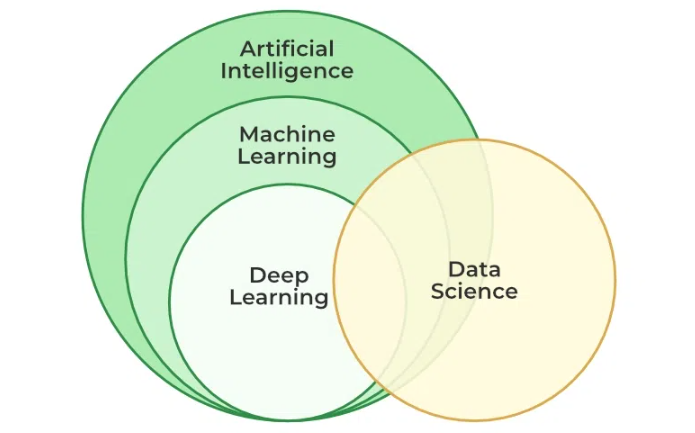
* Radius: Radius is the smallest graph eccentricity; it is the shortest among all maximum distances between any two nodes of a graph.s

**2.Machine Learning**

Machine Learning is a sub-domain of Artificial Intelligence that enables algorithms to recognize patterns implicit in-the data sets and to make forecasts for new, similar data, without being explicitly programed for Projects. Traditional machine learning merges data and statistical tools predicting outputs, producing actionable insights. Applications range as far-flung as image and voice recognition, natural language processing, recommendation systems, fraud detection, portfolio optimization, and automation.

For instance, recommender systems individualize recommendations with previous information. Netflix, for example, will recommend movies and series through collaborative and content-based filtering. Trends of the viewer, ratings provided to a particular movie or series, and favored genres are included in the list. An agent, in conduit with environmental feedback, is thus enabled by reinforcement learning to make decisions, hence improving those recommendations over time.

It affects autonomous vehicles, drones, and robots—making them more adaptive under dynamic environmental situations. The approach has been realized to be key in machine learning, whereby machines are left to learn from examples of data and produce an accurate output inextricably related to data mining and data science.



**2.1. Deep learning**

Deep learning is the subset of machine learning that's basically built around artificial neural networks. And what has been made possible due to deep learning, and I'm sure we might have come across through the recent advancements be it image recognition, or natural language processing, or even autonomous driving for that matter. Basically, the whole idea behind deep learning is it allows us to develop machines like a computer model which can process information with a human being's level of accuracy, so essentially getting equivalent results as with a computer of very high performance alongside your accurate results from natural computers i.e. humans.

**3. Artificial Intelligence**

Artificial intelligence is the replication of human brain activities undertaken by computers or machines. It is an umbrella term applying to different scopes of technologies and methodologies that allow machines—typically computer systems—to perform tasks in ways semblant to human intelligence.

Types of AI:

Narrow AI: Weak AI: The system is designed to perform narrow tasks; for example, recommendation systems, facial recognition working under constrained and limited situations.

General AI: StrongAI:

A truly theoretical form of AI is that would possess the ability to understand and learn intelligence across an extremely wide scope of tasks, not so different from human capabilities.

Superintelligent AI: A form of AI reaching far beyond human intelligence in almost all areas— Purely speculative for now.

Al applications of AIIRQ Healthcare: diagnosis, personalized medicine, medical imaging analysis Finance: fraud-detection, algorithmic-trading, risk management Transport: autonomous vehicles and traffic management systems.

Entertainment: Recommender systems, game AIs, and virtual assistants. Customer Service: Chatbots, automated support.

**4.Python**

Python is a popular interpreted programming language which is highly used to build scalable web applications, data-science and machine-learning projects. The main purpose of the language since its creation was to make code readability by having a clean syntax and structured indentation. It was designed to be really easy to acquire for beginners, but on the other hand, it is also complex enough to serve professional developers too. It's a general-purpose language that can be used in a various fields like in web development, data analysis, artificial intelligence, scientific computing etc.

The capability of the Python comes from libraries included in Python's standard library and over hundreds of thousands external packages provided by these active community. Hence there are many modules and frameworks that speed up the develop application process gray provided by developer resourceful mind sets. Last but not least: it works with other languages & technologies well this among with its flexibility makes it perfect for using in projects combined with other technologies providing certain parts.

**5.Network Analysis**

Network analysis can be defined as a study of interactions between entities in a network. It involves investigating the organization of links or connections between the entities, as well as the properties of the individual entities. Study areas in network analysis range broadly from social to transportation and biological networks.

For example, in social network analysis, the entities are typically individual persons, and the links could represent anything from friendship ties to professional bonds. Configuration of the relationships in a social system could be analyzed so as to identify important influencers of other nodes, the spread of particular kinds of ideas or behavior, or understand the way in which groups are put together.

In the context of transportation networks, the entities would correspond to airports or train stations, and links would correspond to flights or train routes. With network analysis it is possible to find the best routes between places or present bottlenecks and other problems that could make the network suboptimal at peak performance.

Network analysis includes tools and techniques such as graph theory, centrality measures, and methods for visualizing networks. The mentioned tools assist the researcher in identifying patterns and trends in the data and understanding the working structure behind the created network.

**6.NetworkX**

NetworkX is a Python package for creating, manipulating, and studying the structure, behavior, and evolution of complex networks. It gives access to the wide variety of graph and network routines.

**7.Sklearn**

Sklearn is a very popular open-source Python Machine Learning library. The elegance and efficiency of the toolkit, grounded in data mining and analysis, are due to NumPy, SciPy, and Matplotlib.

Common use cases are:

1. Predictive Modeling: A set of techniques that build models from historical data in an attempt to predict outcomes.
2. Data Mining: Exploration of datasets aimed at identifying patterns and relationships.
3. Feature Engineering: Filter or transform features to enhance the performance of the model.

**8.Cosine Similarity**

Cosine Similarity is measure of the similarity between two non-zero vectors in an inner product space, proportional to the cosine of the angle between them. Cosine Similarity finds its application in many spheres, most paramount being text analysis and natural language processing.

Key Features:

Range: The cosine similarity ranges from -1 to 1.

1: Vectors are in the same direction.

0: Orthogonality, indicating no similarity.

-1: Vectors are opposite in direction.

Scale Invariance: It is unaffected by the magnitude of vectors; therefore, it is useful for comparing different-sized documents.

**9.Pickle**

It is a Python module that allows serialization and deserialization of Python objects. This basically implies that conversion of a Python object into byte streams (serialization) and vice-versa, deserialization. Some of the key reasons for using the Pickle library are:

1. Persistent StorageSave Objects: One is able to save complex Python objects like lists or dictionaries or any user-defined class to the disk so easily. This makes saving the program state so easy.

2. Model Persistence Machine Learning Models: It is often used to save the trained machine learning models—let's say, from Scikit-learn—to be loaded in the future without the need for a traverse training loop.

3. Simple to Use and Simple API: The API of Pickle is very easy, and it can be a serialized using pickle.dump() from the object side. It can be deserialized load() from the pickle side. Easy work with this tool.

4. Heterogeneous Data Type Support and Flexibility: An extensive range of Python objects can be pickled, including self-defined classes and complex nested data structures.

5. Efficiency and Performance: In the majority of cases, it is faster than other serialization formats when serializing complex Python objects. Especially it is true for working with a rather big dataset. lattens the input. Does not affect the batch size.

**10. Matplotlib**

One of the favorite packages for charts in Python is Matplotlib. It's a Python 2D plotting library that produces publication-quality figures and provides an easy way to generate plots without high learning curves. It was originally developed by John D. Hunter in 2003 and since then, it has evolved into a its current state that supports many varieties of plots as counting as a user-friendly interface which allows users to create publication quality plots with just few lines of the code. The design of matplotlib is a quite similar to the MATLAB, it makes will make you a feel at home if You are familiar with MatLAB. The library provides several plotting options like line plots, scatter plots; bar charts and histograms;3D plotsetc. This has resulted in many researchers, engineers and finance persons using this for all data visualization needs which are required today by demand.

A significant benefit of using Matplotlib is has these two-levels deep, configurations for a wide range of charting properties. This means changing colors, labels, line types and axes to produce final quality plots. Trigonometry and Calculus using math as core package Data manipulation packages like Numpy, Pandas etc. Visualization libraries such as Matplotlib, line plot. pyplot Implementation is the easy part. It also provides a powerful interactive interface integrated with IPython and Jupyter Notebooks that enables real-time dynamic data exploration and visualization. Additionally, the abundance of documentation on Matplotlib and support from its active community make it even more powerful resource for visualization when compared to other libraries in Python making this a must-use tool for any data science or analysis that you are doing using python.

**11. Sreamlit**

Streamlit is an open source Python library that this user can leverage to effortlessly design web applications for machine learning however Data Science projects. It is used to develop an interactive web application with minimal coding and prioritizes simplicity and ease of use.

Key Features of Streamlit:

* Rapid Development: Quickly convert Python scripts into web apps without deep knowledge in web development.
* Interactive Widgets: It comes with many built-in widgets, including sliders, buttons, and text inputs.
* Real-Time Updates: Its app can update in real-time and provide users with the results in real-time, making changes visible to a users instantly.
* Easy Integration: It facilitates easy integration with other widely used data science libraries like NumPy, Pandas, Matplotlib, and Plotly.
* Ease of Deployment: It is easy to deploy Streamlit apps on Streamlit Sharing, Heroku, or any other web server.
* Markdown support: It presents proper formatting of text, allows Markdown, which makes documentation and explanation add easily.

**2.5 HARDWARE AND SOFTWARE REQUIREMENTS**

**HARDWARE REQUIREMENTS**

* CPU - Intel i3 and above recommended.
* RAM – 8GB.
* Storage – 256 HDD/SSD.
* GPU – RTX 1050 Ti or higher recommended.

**SOFTWARE REQUIREMENTS**

* Operating Syatem - Windows 10 or above/ ubuntu 20.04 or above.
* Coding Language: Python 3.10 or above
* Web Framework: Streamlit

**CHAPTER 3**

**SOFTWARE REQUIREMENTS SPECIFICATION**

**3.1 FUNCTIONAL REQUIREMENTS**

**Data Gathering:** Data gathering refers to a process of collecting, compiling, and assembling information or data from various sources so that it can generate accurate and reliable information.

* **Data Preprocessing:** Includes Before any analysis, the data may have to be cleaned and formatted. This may involve sorting, cleaning missing or misvalued data, and data aggregation if required. Making it suitable for accurate analysis.
* **Data visualization:** This kind of data is usually hard to understand in most instances, especially when it is a large or complex network. Diagraphy can be of great help in understanding the overall structure of a network, hence enabling the detection of patterns or trends. These comprise diverse tools and techniques for network data visualization, like diagrams, node-link diagrams, and matrix plots.
* **Statistical analysis:** Once the data of the network has been prepared and visualized, next comes statistical techniques with the network data. It could be measures of centrality or community structures, testing for the correlation of different variables, or model fitting.
* **Interpretation:** Network analysis interprets the results of analysis and makes conclusions based on the network. This may consist of answering the research questions, testing hypotheses, or prediction in the network.

**3.2 NON-FUNCTIONAL REQUIREMENTS**

* **Scalability -** To ensure that the movie recommendation system can effectively handle increasing amounts of data, users, and requests without compromising performance.

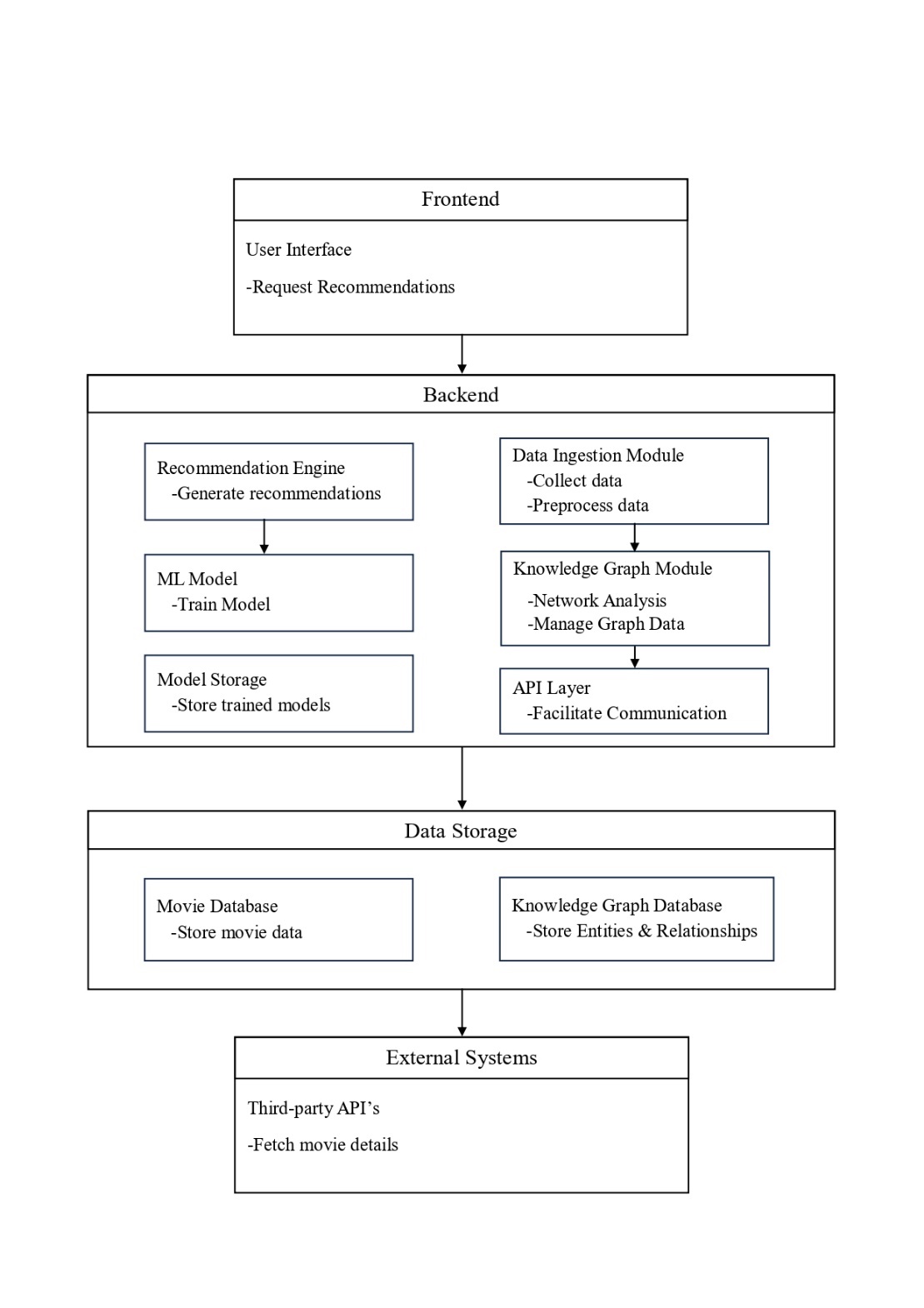
Scalability is critical for the movie recommendation system to ensure it can adapt to growing datasets, increasing user numbers, and evolving application demands while maintaining performance and user satisfaction. A well-designed scalable architecture will facilitate future growth and feature expansion.

* **Reliability -** The system must be reliable to provide accurate results.
* **Usability -** The system must be user-friendly helping users to interact with the system easily. As usability plays an important role for the user to make use of the system to a full extent.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 SYSTEM PERSPECTIVE**



*The fig 4.1.1*

A system architecture diagram for a movie recommendation-system that makes use of knowledge graphs and machine learning would entail illustrating the different components and how they relate to one another. Here is the detailed explanation of the architecture components are:

Frontend User Interface:- The user interacts with the system to get recommendations and rate movies.

Backend:-

Recommendation Engine: This holds the core logic for generating movie recommendations.

Knowledge Graph Module: This manages and queries the knowledge graph.

Machine Learning Module: It trains the model against the training data and makes predictions.

Data Ingestion Module: handles data collection and preprocessing.

API Layer: It exposes an interface through which the frontend can talk to the backend.

Data Storage:-

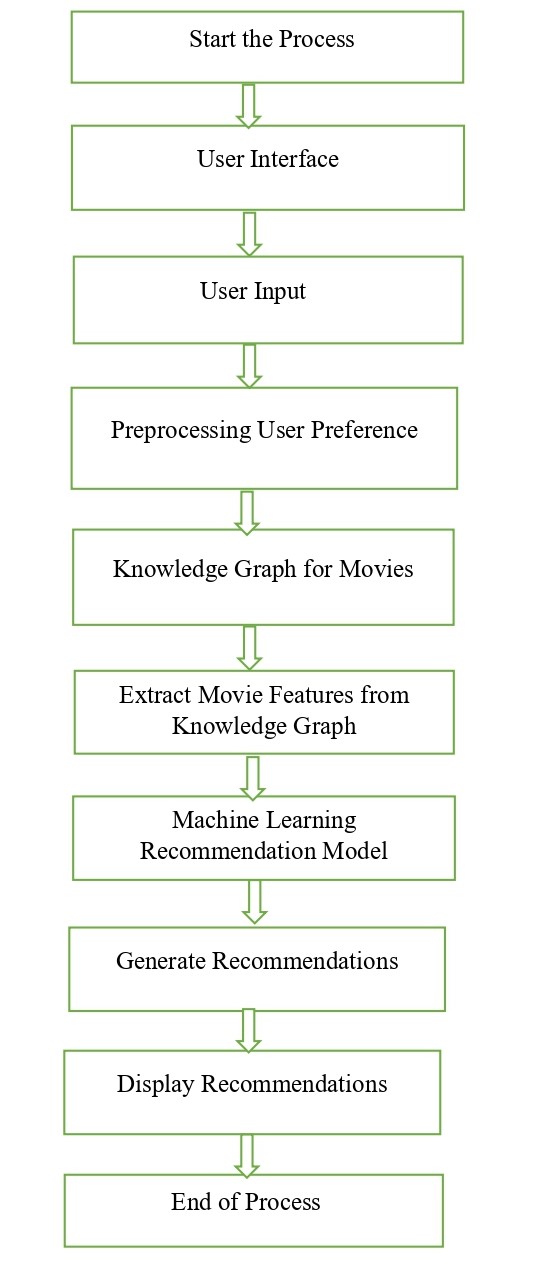
Movie Database: Info regarding movies, users, and their ratings.

Knowledge Graph Database: This will be used to store entities and relationships.

External Systems:-

Third-Party APIs: retrieves supplementary information, such as those on movies.

**4.2 DATA** **FLOW DIAGRAM(DFD)**

To design a data flow diagram for a movie recommendation system Guzzle based on knowledge graph and machine learning, an account of where the data is supposed to pass through and at which phase processing occurs has to be put into consideration.

*The fig 4.2.1*

User Interface: It is the interface through which a user interacts with the system.

User Input: Input by the user in terms of ratings, genre preferences, or specific titles of movies.

Preprocessing User Preferences: A preprocessing step to normalize and structure the user preferences for further processing.

Query Knowledge Graph for Movies: Querying the knowledge graph to obtain those movies that will best fit the preferences or criteria provided by the user.

Extract Movie Features from Knowledge Graph: Extraction of relevant features of movies from the knowledge graph, such as genres, actors, directors, dates of release, etc.

Machine Learning Recommendation Model: The core machine learning model responsible for generating movie recommendations based on extracted features and user preference.

Generate Recommendations: Generating a list of recommended movies based on the output of the recommendation model.

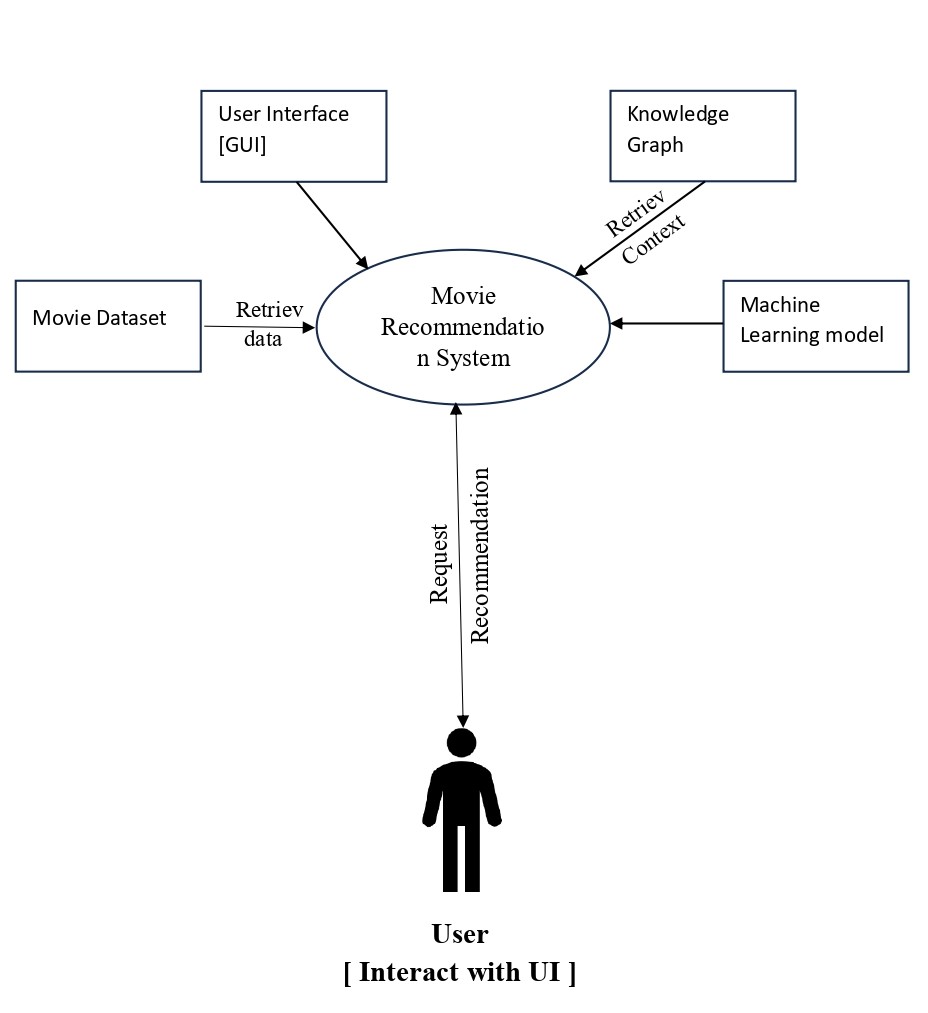
Display Recommendations: Displaying the recommended movies to the user through the user interface.

**CHAPTER 5**

**DETAILED DESIGN**

**5.1 CONTEXT DIAGRAM:**

The context diagram performs the whole system overview and interaction between the system and entities external to it. Typically, it is a system with one process, along with its interactions among the external actors and systems.

****

*The fig. 5.1.1*

**5.2.** **USE CASE DIAGRAM:**

A use case diagram of a movie recommendation system, both knowledge graph- and machine learning-based, shows the interaction between the actors and the system, emphasizing the numerous use cases within the system. Here is how we can structure it:

Actors:-

User: The person who wants recommendations.

Admin: Administers and updates the system with respect to the knowledge graph, and model training.

Use Cases:-

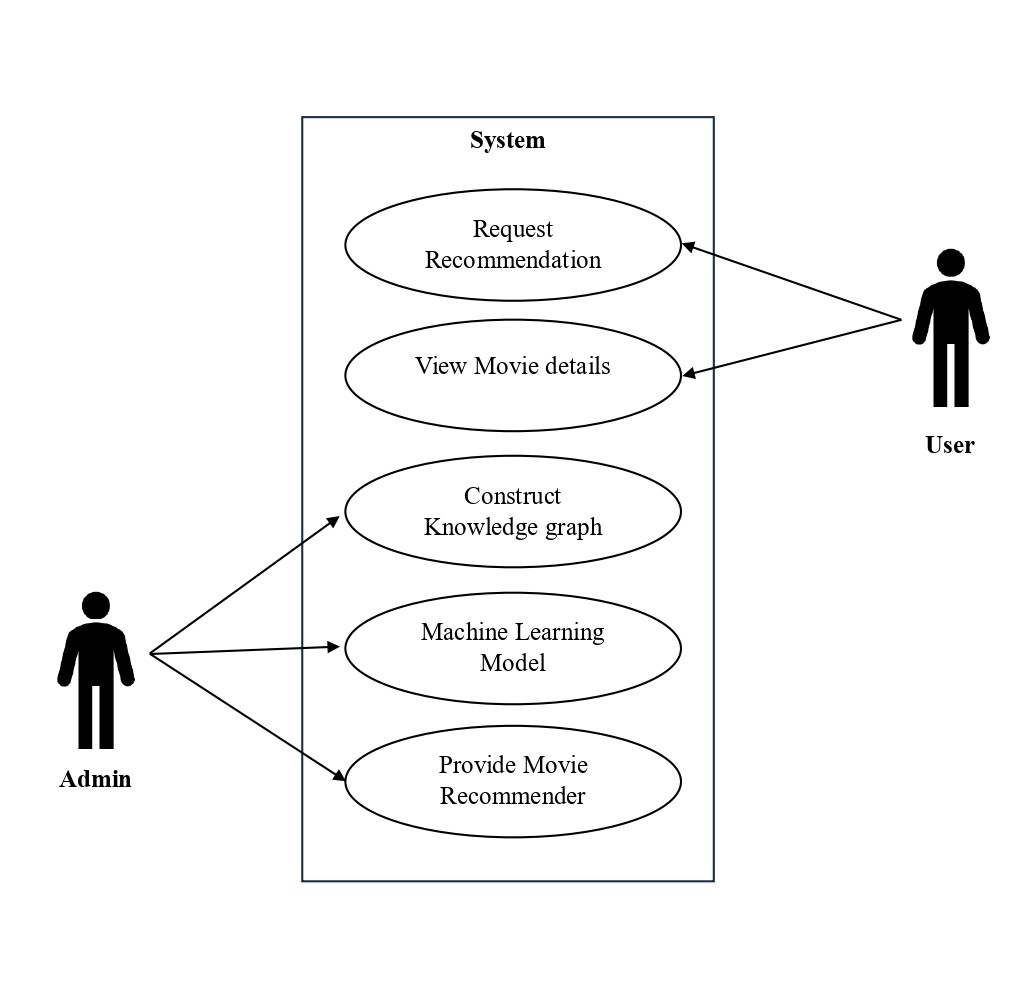
Request Recommendation: Request recommendations.

Viewing Movie Details: The user views full information about the movie.

Construct knowledge graph: Admin may update the knowledge graph with new data.

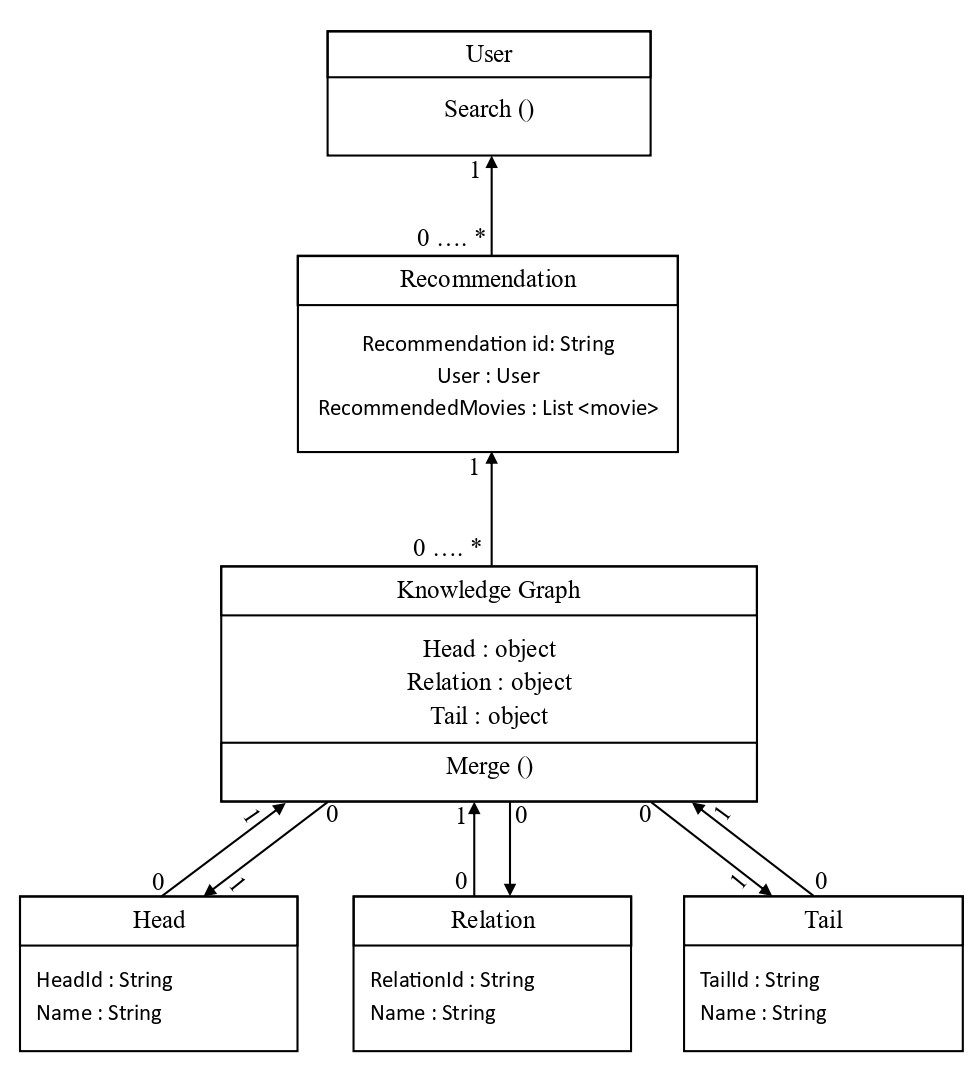
Machine Learning Model: Admin trains the machine learning model on the data.

Provide Movie Recommender: Admin manages user data and provide recommendations.

*The fig. 5.2.1*

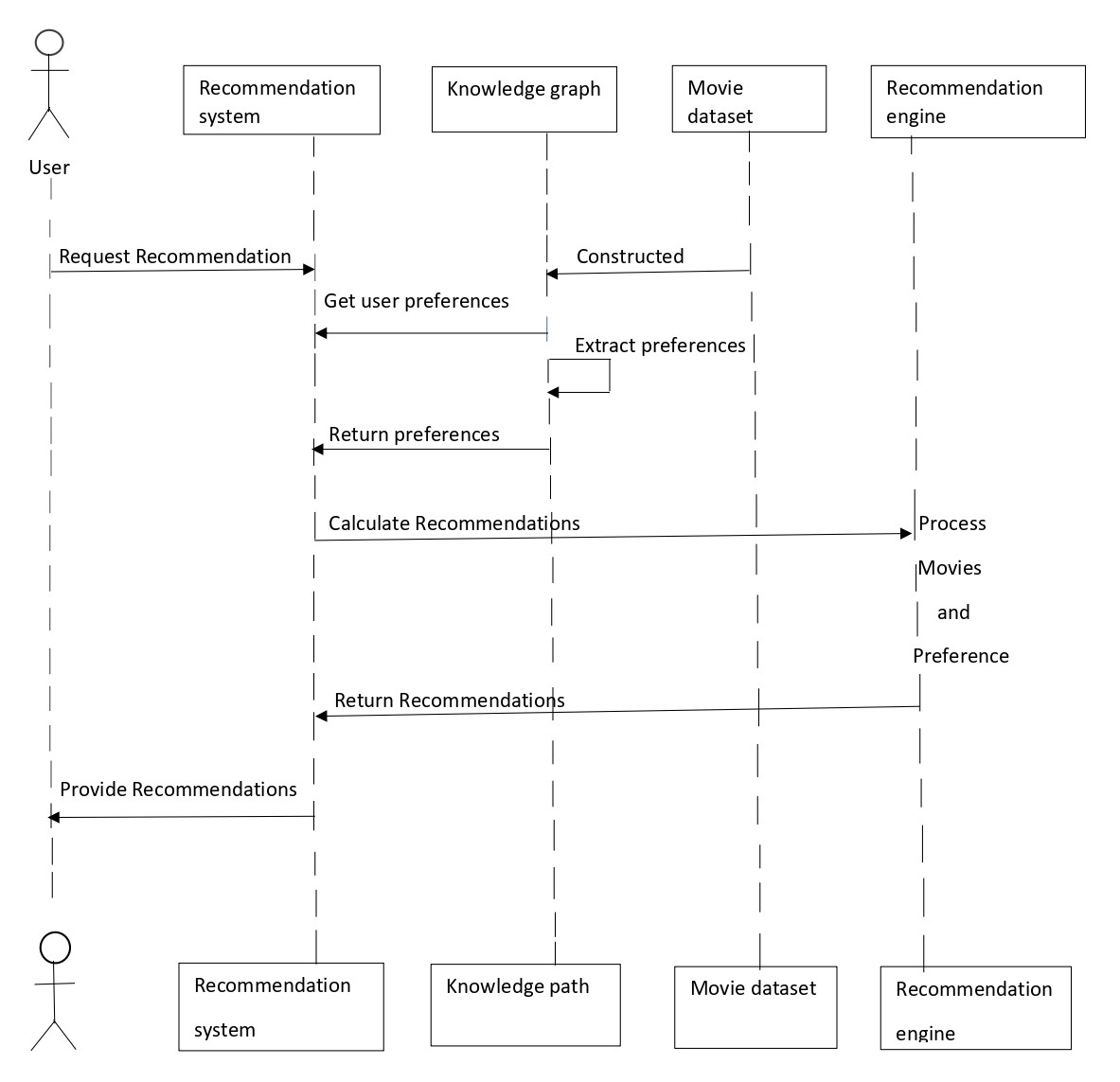
This use case diagram should illustrate the main interactions between the movie recommendation system and its users in terms of major functionality and system roles.

**5.3.** **CLASS DIAGRAM:**

****A class diagram for a movie recommendation system basically based on a knowledge graph and machine learning will identify key components and their relations.

*The fig 5.3.1*

**5.4 SEQUENCE DIAGRAM**

**** This is the sequence diagram of a movie recommendation system using knowledge graphs and machine learning

*The fig 5.4.1*

**CHAPTER 6**

**IMPLEMENTATION**

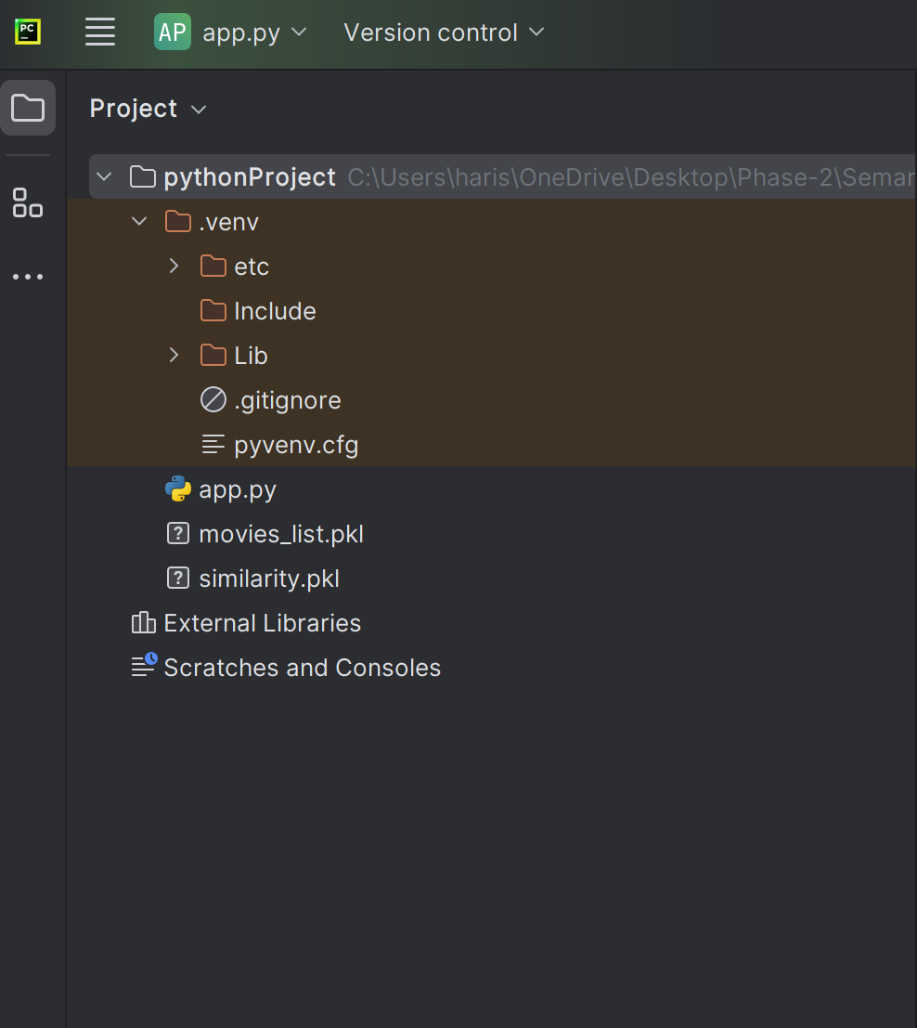
**Folder Structure**

Fig 6.1.1 - Folder structure

The above figure shows how the files and folders are organized to get the actual requirement done, and dividing the project into multiple structured directories will help us to easily track and maintain the project. Now, let's dive deep into the structure of the project.

Herein, all the directories and files are arranged under a folder called PYTHON\_PROJECT, which serves as the parent folder for this project.

**App.py** – This is a Python file responsible for serving the ML model to the end user by making use of API services. Basically, this is the Flask backend file that handles any form of request from the user having to do with this particular application and serves an appropriate result back to the end user.

**.gitignore –** This will be a GitHub file wherein we need to only mention the names of files and directories which we do not intend to be pushed to the GitHub. As soon as the file names get updated here, all of these files or directories will then be ignored by this very version control system while uploading the code to the GitHub.

**movies\_list.pkl** – This could be an object of Python's list type, containing information related to movies, such as movie titles and genres. We normally need to load the data back into Python from a .pkl file using the function pickle.load() from the module pickle.We would use pickle.dump() to serialize your Python object and then write it out to a file to create a .pkl file.

**Similarity.pkl**– Similarity.pkl can be a serialized file. It would probably store a similarity matrix or even a model related to the computation of similarity. This could be applied in recommendation systems or any other data-driven applications aimed at computing similarities between different items, based on a set of features or interactions.

**6.1 MODULES:**

1. Data Collection
2. Dataset
3. Data Preparation
4. Network Analysis
5. Machine Learning Model

**MODULES DESCSRIPTION:**

**1.Data Collection:**

Data collection refers to the process of gathering and measuring information on variables of interest in an organized way. Gathering and compiling data are essential parts of research, analysis, and decision-making in many disciplines, directly or indirectly related to science, social sciences, business, and technology.

Kaggle Link: <http://www.kaggle.com/datasets/gargmanas/movierecommenderdataset>

**2.Dataset:**

df This will be the variable name that holds the result DataFrame. In Python, it's pretty standard to use df as shorthand for "DataFrame".

pd : This is an alias for the pandas library. You would generally put the import statement for the pandas library at the top of your script: import pandas as pd in Python. The as pd part creates an alias called pd for the pandas library so that when you want to use any functions or classes from the library, you can simply use pd.

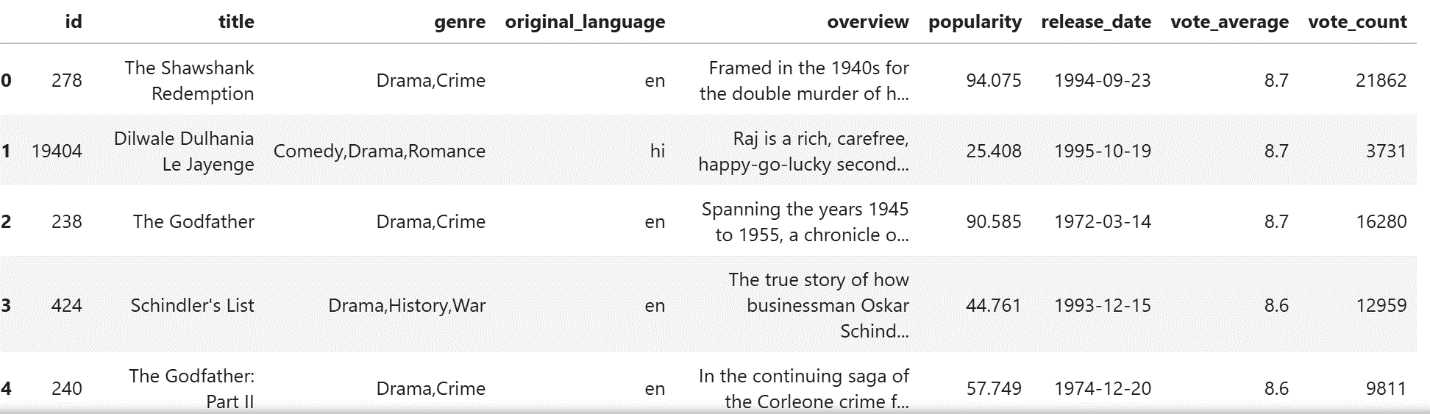
read\_csv—One of the pandas library functions that reads a Comma Separated Values, CSV, file into a DataFrame. The read\_csv function represents a fairly robust but somewhat flexible way to import data from a text file into a Pandas DataFrame'.

r'C:\Users\archive\dataset.csv' is a directory leading to the CSV file, which will be read into the DataFrame. What is going on here is:

The 'r' actually is a prefix indicating a raw string literal. It's how you instruct Python to treat the string as raw, with no special character interpretation. This is often quite useful when dealing with a Windows file paths, especially where backslashes (\) are usually found.

C:\\Users\\archive\\dataset.csv is just the real path to your CSV file. This path is system-specific—Windows—and this 'r' prefix makes it right.

df=pd.read\_csv(r'C:\\Users\\archive\\dataset.csv') reads from a CSV file at C:\\Users\\archive\\dataset.csv into a Pandas DataFrame. This resulting DataFrame is assigned to the variable df.



**3.Data Preparation:**

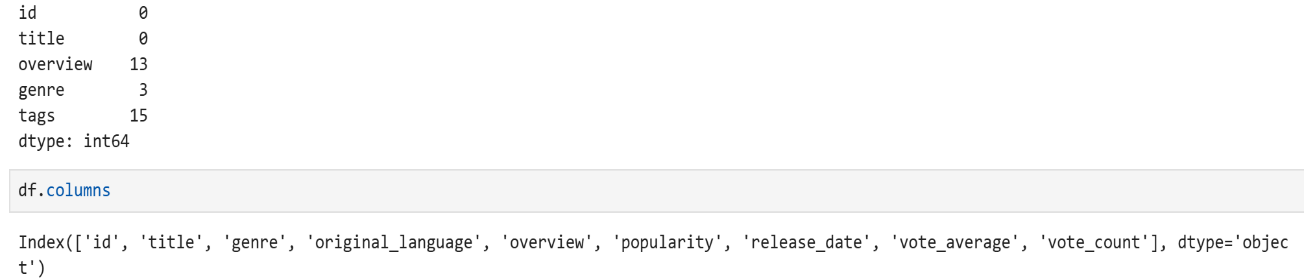
Data preparation, or preprocessing of data, is an exceedingly important stage in the pipeline of any analysis of data. It comprises translation of raw data into a form that will let it be analyzed and modeled. At this step, it will be ensured that prepared data are clean, organized, and relevant for any analysis to be performed efficiently and effectively.

3.1 Data Cleaning:

It will help in detecting and counting the null values in a given Pandas DataFrame. movies.isnull().sum() Let's break this down:

Movies:This is what you have called that Pandas DataFrame that's holding your data. In this case, it would probably turn out to be a DataFrame containing movie information.

isnull(): It is from Pandas and will output a Boolean mask in such a way that corresponding Boolean value will denote whether the value in the DataFrame is any Null/missing. It is somewhat similar to an if condition for checking Null values.

sum(): This is a method through which all the True values, obtained from the resulting Boolean mask, are summed.

3.2 Data Integration:

Combining required columns into one single column.

title', 'overview', 'genre' are columns we want to select from the original DataFrame. To selection of three columns such as title, overview and genre.

When we glue everything together, the code df = df[[ 'title', 'overview', 'genre']] does the following:It selects only columns title, overview, and genre from the original DataFrame df. This would reassign the resulting DataFrame back to the original variable df, which includes only the selected columns.



**4.Network Analysis:**

Network analysis, sometimes referred to simply as network or graph theory, is a subfield of mathematics and computer science that deals with the study of structure, properties, and dynamics of networks. It provides insight into the heterogeneous relationships and interactions of nodes (usually entities) connected by edges in data science and social sciences.

def label\_to\_id(df):

# we use set to make our list of entities/relations unique

# set is an unordered

entities = set(df['head'].tolist())

entities.update(df['tail'].tolist())

relations = set(df['relation'].tolist())

print(f'There are {len(entities)} entities and {len(relations)} relations.')

entities = set(str(entity) for entity in entities)

relations= set(str(relation) for relation in relations)

#label\_dict = {int: string} for which int holds the entity id while string holds the original label

label2id\_dict = {label: idx for idx, label in enumerate(sorted(entities))}

# extend label\_dict to also include relations

for idx, label in enumerate(sorted(relations)):

# assuming that we have 10 entities, then idx assigned to relations will start from 10 onwards

label2id\_dict[label] = idx+len(entities)

#Now let's convert our original triples to store ids instead of strings

triples = []

# we use label2id\_dict to retrieve the assigned id given the entity name

for index, row in df.iterrows():

triples.append((label2id\_dict[row['head']],label2id\_dict[row['relation']], label2id\_dict[row['tail']]))

print(f'There are {len(triples)} triples.')

return (label2id\_dict, triples)

label2id\_dict, triples = label\_to\_id(df)

O/P: There are 19629 entities, 2123 relations and there are 9985 triples.

4.1 Knowledge Graph Construction:

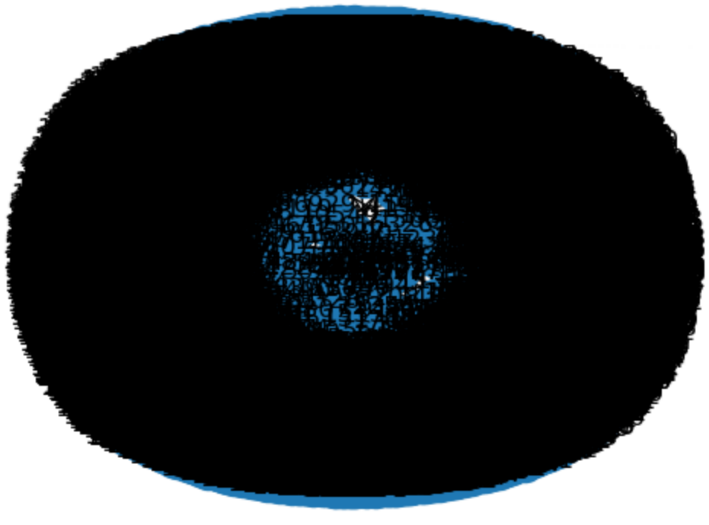
#Visualization

import matplotlib.pyplot as plt

%matplotlib inline

pos = nx.spring\_layout(G, k=0.05, seed=42)

nx.draw\_networkx(G, pos, with\_labels = True)



4.2 Plotting Subgraph:

It will show the graph with node labels by ids.

def draw\_subgraph(subgraph\_G, with\_labels=False):

pos = nx.spring\_layout(subgraph\_G, seed=333)

plt.figure()

node\_labels = {}

edge\_labels = {}

if with\_labels:

# retrieve labels from id2label\_dict

node\_labels = {node: id2label\_dict[node] for node in subgraph\_G.nodes()}

edge\_labels = {(h, t): id2label\_dict[r['label']] for h, t, r in subgraph\_G.edges.data()}

else:

node\_labels = {node: node for node in subgraph\_G.nodes()}

edge\_labels = {(h, t): r['label'] for h, t, r in subgraph\_G.edges.data()}

# draw the nodes

nx.draw(subgraph\_G, pos, edge\_color='black', width=1, linewidths=1,

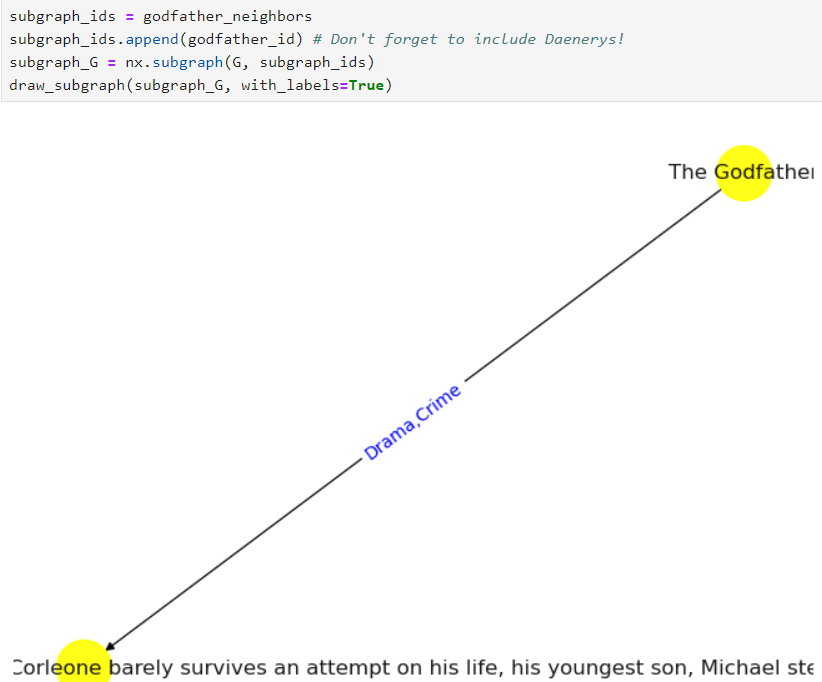
node\_size=1000, node\_color='yellow', alpha=0.9,labels=node\_labels )

# take the edge\_labels

nx.draw\_networkx\_edge\_labels( subgraph\_G, pos, edge\_labels=edge\_labels,

font\_color='blue')

plt.axis('off')

 plt.show()

**5.Machine Learning Model:**

A model in machine learning is a mathematical representation or algorithmic structure that learns from data. It is designed to make predictions, classification, or decisions for input data without being explicitly programmed for specific tasks.

from sklearn.feature\_extraction.text import CountVectorizer

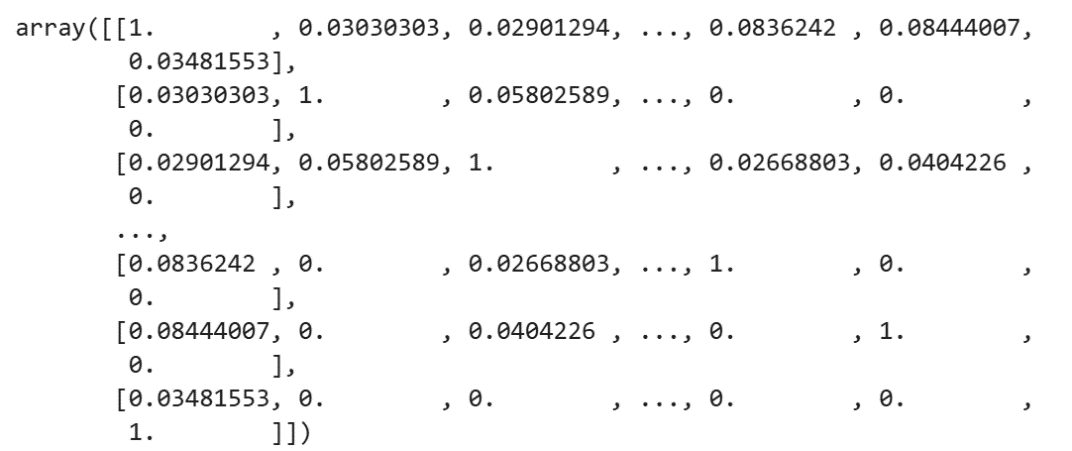
from sklearn.metrics.pairwise import cosine\_similarity

cv=CountVectorizer(max\_features=10000, stop\_words='english')

vector=cv.fit\_transform(df['tail']).toarray()

similarity=cosine\_similarity(vector)

Similarity.



def recommand(movies):

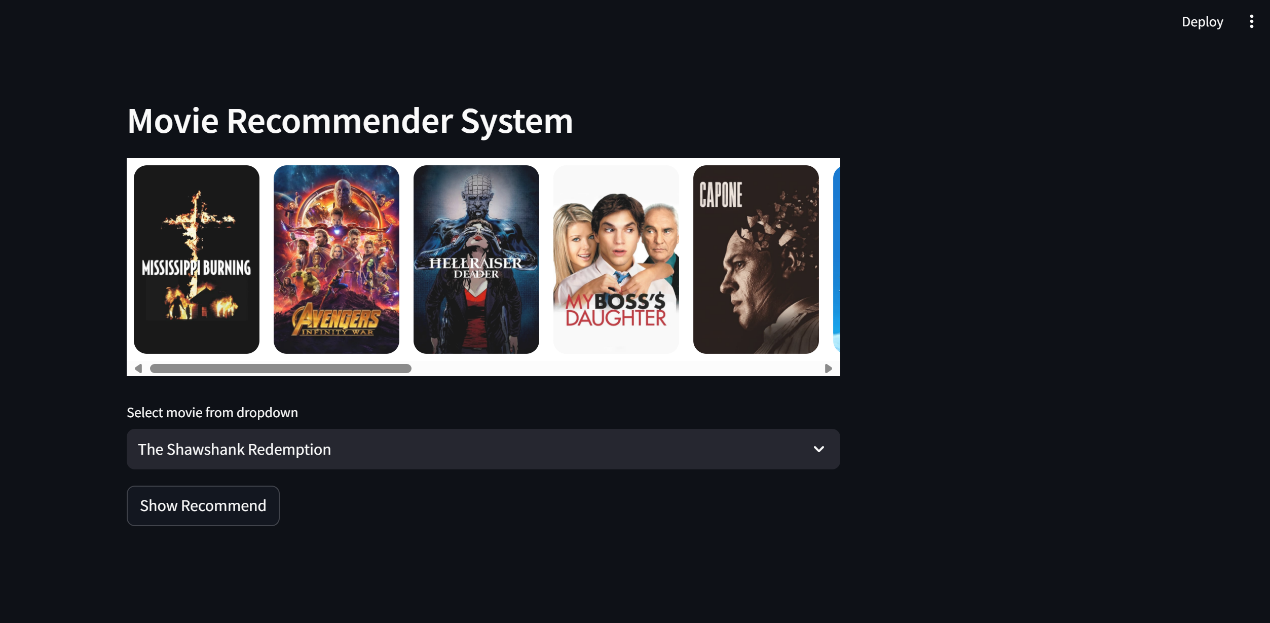
index=new\_data[df['head']==movies].index[0]

distance = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda vector:vector[1])

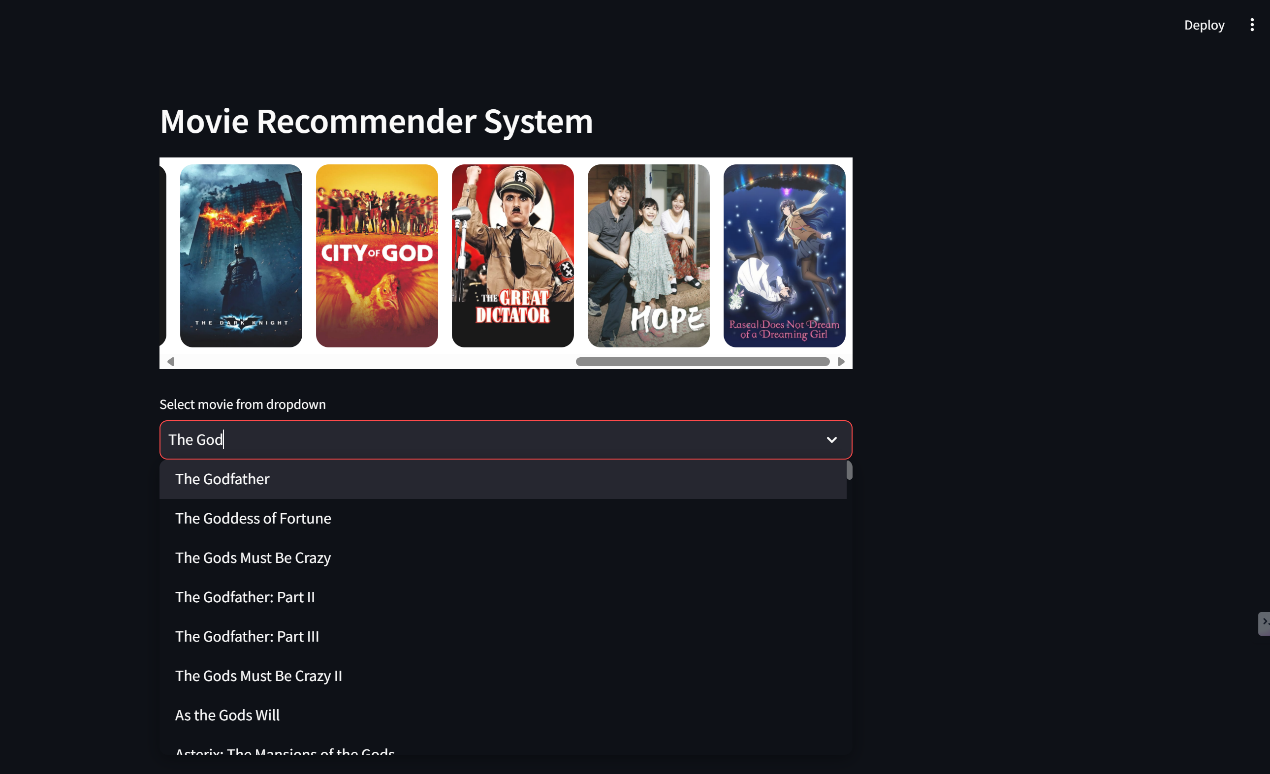
for i in distance[0:5]:

print(df.iloc[i[0]].head)

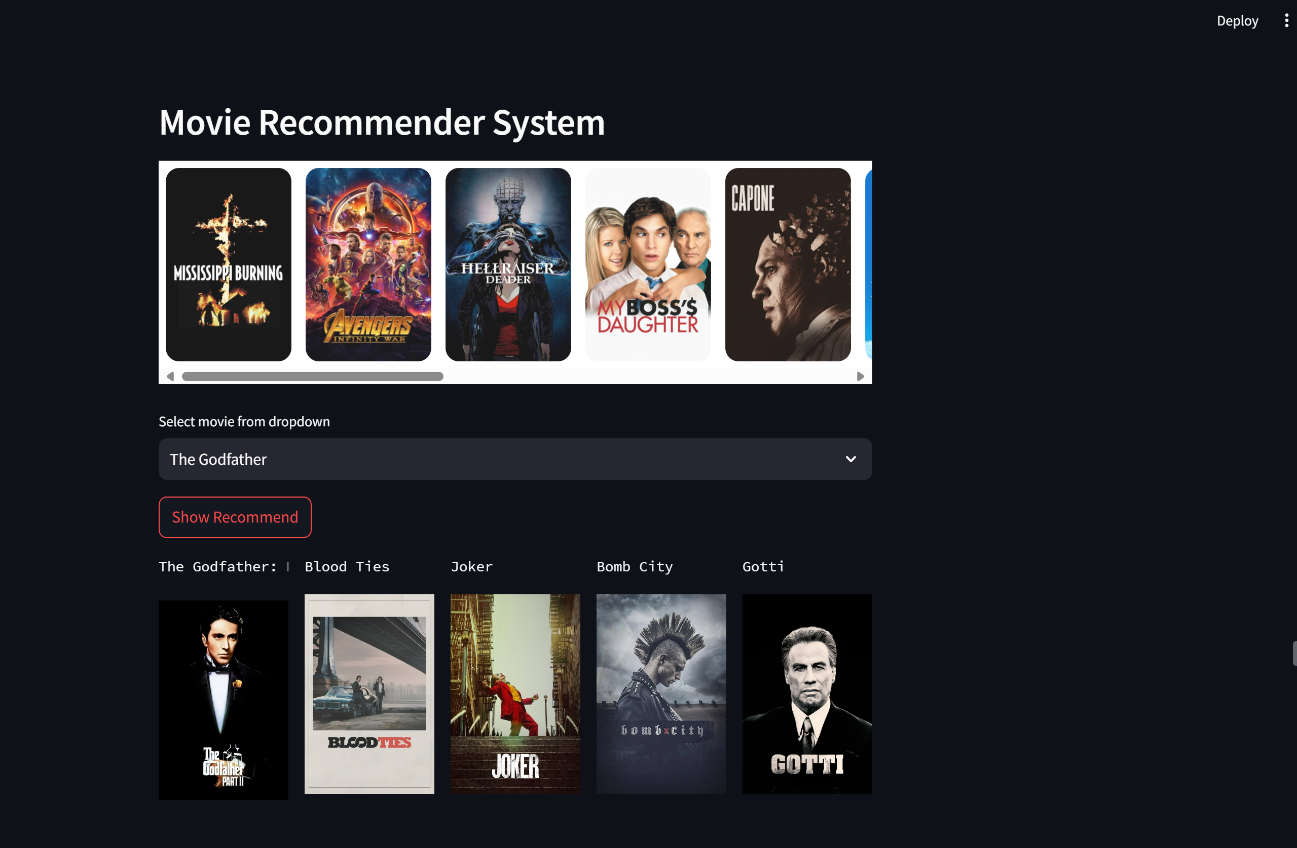
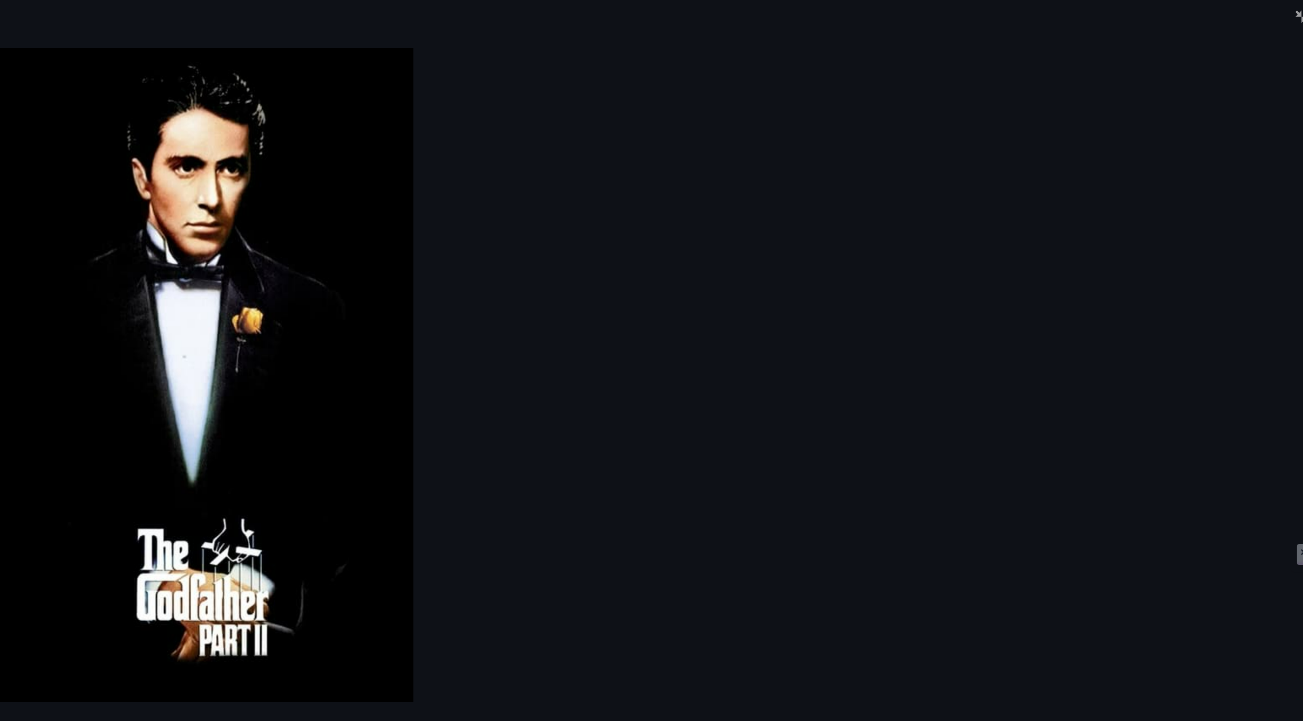
recommand("The Godfather")

**6.2** **Screenshots:**

6.2.1 Home Page

This is the landing page for the user where they can search the movie.

6.2.2 User Searching Page



6.2.4 Review Of Movie

6.2.3.Movies Recommended Based on User Search

6.2.3 Prediction page showing no sign of disease is detected

**CHAPTER 7**

**SOFTWARE TESTING**

Software testing is a major process the in the software development life cycle to find the errors or bugs in the developed software and to ensure that the developed software is functioning properly.

**Manual testing**

Manual testing is a type of testing the working of model by giving the different kinds of inputs to the machine manually to check whether it will work correctly or not.

**Table 7.1: Test Case: Check Recommendations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Scenario** | | **Check Recommendations** | | |
| **Test case ID** | **Step details** | **Expected result** | **Actual results** | **Pass/Fail/Not-Executed/Suspended** |
| TC01 | Navigate to http://localhost:8501 | Site should open | Site Opened as expected | PASS |
| TC02 | Select Movie Form Dropdown | Selected Movie Related Movie Recommends | Selected Movie Related Movies Recommends | PASS |
| TC03 | Without Selecting a Movie , click | It does not offered any button, Without select | It does not offered any button, Without select | PASS |
| TC04 | By search text, in search bar | It displays list of movies related to text what user trying to search | It displays list of movies related to text what user trying to search | PASS |
| TC05 | If in case searched text is not related to current list | Displays no results | Displays no results | PASS |

**CHAPTER 8**

**CONCLUSION**

A movie recommendation system using knowledge graphs and machine learning would be a robust and sophisticated approach to personalized content delivery. In the proposed dual-method strategy, it will improve upon the recommendation quality by combining rich context from knowledge graphs with inductive power from machine learning algorithms.

It not only improves user satisfaction but also maintains the elasticity and no-end point tendency of the system, able to evolve with new data and user preference.

**CHAPTER 9**

**FUTURE ENHANCEMENT**

Other improvements will involve increasing the datasets to be more diverse and optimizing the models to process in real-time. Integration into various other fields will make them much more useful. Semantic Web Technology is a phenomenal improvement of the representation, sharing, and usage of data on the web. It provides a framework that may tell machines what information means and thereby assists in the development of more intelligent, more efficient data processing for developing smarter applications and more interconnected web resources.

**APPENDIX A**

**BIBLIOGRAPHY**

1. ANNA BREIT, LAURA WALTERSDORFER, FAJAR J. EKAPUTRA, MARTA SABOU, ANDREAS EKELHART, ANDREEA IANA, ARTEM REVENKO, ANNETTE TEN TEIJE and FRANK VAN HARMELEN.

"Combining Machine Learning and SemanticWeb: A Systematic Mapping Study".

<https://doi.org/10.1145/3586163>

1. . Sarah Ghidalia, Ouassila Labbani Narsis, Aurélie Bertaux, Christophe Nicolle.

"COMBINING MACHINE LEARNING AND ONTOLOGY: A SYSTEMATIC LITERATURE REVIEW".

<https://www.w3.org>

1. Artem Revenko(Semantic Web Company).

"Supplementary Objectives: Analysing The Motivations Behind SemanticWeb Machine Learning System Design".

The First Austrian Symposium on AI, Robotics, and Vision (AIROV24)

1. Arne Seeliger, Helmut Krcmar, Matthias Pfaff.

"Semantic Web Technologies for Explainable Machine Learning Models: A Literature Review"

<https://www.researchgate.net/publication/336578867>

1. Nikos Kanakaris, Nikolaos Giarelis, Ilias Siachos and Nikos Karacapilidis.

"Shall I Work with Them? A Knowledge Graph-Based Approach for Predicting Future Research Collaborations"

<https://doi.org/10.3390/e23060664>

1. Yufeng Li, Weimin Wang, Xu Yan, Min Gao, and MingXuan Xiao.

"Research on the Application of Semantic Network in Disease Diagnosis Prompts Based on Medical Corpus". <https://doi.org/10.55524/ijircst.2024.12.2.1>

1. Rowida Alfrjani, Taha Osman ∗, Georgina Cosma.

"A Hybrid Semantic Knowledgebase-Machine Learning Approach for OpinionMining" [www.elsevier.com/locate/datak](http://www.elsevier.com/locate/datak)

1. DingJu Zhu, LianZi Xie, BingXu Chen, JianBin Tan, RenFeng Deng et al.

"Knowledge graph and deep learning based pest detection and identification system for fruit quality" [www.sciencedirect.com/journal/internet-of-things](http://www.sciencedirect.com/journal/internet-of-things)

1. Yuexin Huang, SuihuaiYu, JianjieChu1, Zhaojing Su, Yangfan Cong, HanyuWang and Hao Fan. "Combining Deep Learning with Knowledge Graph for Design Knowledge Acquisition in Conceptual ProductDesign"
2. Marko Horvat, Andrija Krtali´c, Amila Akagi´c and Igor Mekterovi´c.

"Ontology-Based Data Observatory for Formal Knowledge Representation of UXO Using Advanced Semantic Web Technologies"

<https://doi.org/10.3390/electronics13050814>

1. Huimin Luo, Weijie Yin, Jianlin Wang, Ge Zhang, Wenjuan Liang, Junwei Luo, and Chaokun Yan.

"Drug-drug interactions prediction based on deep learning and knowledge graph: A review"

<http://creativecommons.org/licenses/by-nc-nd/4.0/>

1. Youtube(<https://www.youtube.com/watch?v=gtGKkBFuvlw&list=PLNXdQl4kBgzubTOfY5cbtxZCgg9UTe-uF&index=13>)

**APPENDIX B**

**USER MANUAL**

**Step 1 :** Open Jupyter Notebook and Pycharm IDE.

**Step 2:** Install Streamlit package in terminal (pip install stearmlit).

**Step3:** Run Program in terminal (Streamlit run app.py)

**Step 4:** If once program will run without errors that directly connect through our favorite browser and navigate to <http://localhost:8501> 1or <http://192.168.104:8501>

**Step 5:** We will search the movies in a web page.

**Step 6:** If user searched movie is available in list, that will provide the best related movies.

**Step 6:**Finally review the specific movie.