UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Enhancing Recommendation Systems with Large Language Models for Accuracy and Explainability

Diploma Thesis

Anastasios Koutsogiannopoulos

Supervisor: Michael Vassilakopoulos

July 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Enhancing Recommendation Systems with Large Language Models for Accuracy and Explainability

Diploma Thesis

Anastasios Koutsogiannopoulos

Supervisor: Michael Vassilakopoulos

July 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ενίσχυση των Συστημάτων Συστάσεων με Μεγάλα Γλωσσικά Μοντέλα για Ακρίβεια και Επεξηγησιμότητα

Διπλωματική Εργασία

Αναστάσιος Κουτσογιαννόπουλος

Επιβλέπων: Μιχαήλ Βασιλακόπουλος

Ιούλιος 2025

Approved by the Examination Committee:

Supervisor **Michael Vassilakopoulos**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Aspassia Daskalopulu**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Eleni Tousidou**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I would like to thank my supervisor, Michael Vassilakopoulos, for his continuous guidance, valuable feedback, and steady support throughout this thesis. His expertise and encouragement were crucial at every stage of the process.

I am also thankful to my friends for turning hard days into lighter ones and making this whole experience something I will look back on with a smile.

Above all, I am grateful to my family for their quiet strength, patience, and constant belief in me during this long journey.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Anastasios Koutsogiannopoulos

Diploma Thesis

Enhancing Recommendation Systems with Large Language Models for Accuracy and Explainability

Anastasios Koutsogiannopoulos

Abstract

Recommender systems are widely used to offer personalized suggestions across digital platforms. However, they still face important limitations related to prediction accuracy and transparency. This thesis addresses these challenges by exploring how Large Language Models can be effectively integrated into the recommendation process.

We initially propose a semantic recommendation model based on Sentence-BERT, which encodes textual metadata into dense vector representations. Evaluation results show that this model consistently outperforms traditional collaborative and content-based methods.

To further improve predictive performance, we develop a hybrid model that combines SBERT with user-based and item-based collaborative filtering. Using XGBoost as a meta-learner, the system learns to weigh the individual predictions and estimate user preferences.

Beyond accuracy, we focus on explainability by fine-tuning a LLaMA 3 (7B) model. This system generates natural-language justifications for recommendations by grounding its output in items similar to those the user has previously seen and liked.

Finally, to enable more interactive and adaptive recommendations, we present a conversational recommender system that incorporates semantic search, intent detection, dialogue tracking, and LLM-based generation. The assistant interprets user input, adjusts its suggestions, and maintains coherence across turns.

Our results on the MovieLens 1M dataset demonstrate that integrating LLMs into both the recommendation pipeline and the explanation layer contributes to user trust, interaction fluency, and a more engaging user experience, confirming their role in next-generation, human-centric recommender systems.

Keywords:

Recommender Systems, Large Language Models, Explainability, Hybrid Models, Sentence-BERT, XGBoost, LLaMA 3, Conversational AI

Διπλωματική Εργασία

Ενίσχυση των Συστημάτων Συστάσεων με Μεγάλα Γλωσσικά Μοντέλα για Ακρίβεια και Επεξηγησιμότητα

Αναστάσιος Κουτσογιαννόπουλος

Περίληψη

Παρόλο που τα συστήματα συστάσεων χρησιμοποιούνται ευρέως, εξακολουθούν να αντιμετωπίζουν προκλήσεις στην ακρίβεια και τη διαφάνεια. Η παρούσα διπλωματική εργασία διερευνά πώς τα Μεγάλα Γλωσσικά Μοντέλα (LLMs) μπορούν να ενισχύσουν τόσο την πρόβλεψη όσο και την επεξήγηση των συστάσεων.

Αρχικά, προτείνεται ένα σημασιολογικό μοντέλο βασισμένο στο Sentence-BERT,το οποίο μετατρέπει βασικά κειμενικά στοιχεία σε διανυσματικές αναπαραστάσεις. Η αξιολόγηση δείχνει ότι το μοντέλο αυτό υπερτερεί σε σχέση με παραδοσιακές μεθόδους συνεργατικού (CF) και βασισμένου στο περιεχόμενο φιλτραρίσματος (CBF).

Στη συνέχεια, αναπτύσσεται ένα υβριδικό μοντέλο που συνδυάζει το SBERT με συνεργατικά μοντέλα, user-based και item-based CF, χρησιμοποιώντας τον XGBoost ως meta-learner για τη στάθμιση των επιμέρους προβλέψεων.

Για την παραγωγή κατανοητών εξηγήσεων, εφαρμόζεται fine-tuning σε ένα μοντέλο LLaMA 3 (7B), το οποίο παράγει κείμενα φυσικής γλώσσας που αιτιολογούν τις προτάσεις βάσει παρόμοιων ταινιών που έχει δει ο χρήστης.

Τέλος, υλοποιείται ένας συνομιλιακός βοηθός συστάσεων που ενσωματώνει σημασιολογική αναζήτηση, αναγνώριση πρόθεσης, διαχείριση διαλόγου και παραγωγή λόγου μέσω LLM. Ο βοηθός ανταποκρίνεται στις ανάγκες του χρήστη σε πραγματικό χρόνο και προσαρμόζει τις προτάσεις του με βάση τη ροή της συζήτησης.

Η αξιολόγηση στη βάση δεδομένων MovieLens 1M δείχνει ότι η αξιοποίηση των LLMs βελτιώνει την ακρίβεια, την κατανόηση και τη διαδραστικότητα, συμβάλλοντας σε πιο αξιόπιστα και φιλικά προς τον χρήστη συστήματα συστάσεων.

Λέξεις-κλειδιά:

Συστήματα Συστάσεων, Μεγάλα Γλωσσικά Μοντέλα, Επεξηγησιμότητα, Υβριδικά Μοντέλα

Table of contents

List of figures

List of tables

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BERTScore	BERT-based Semantic Similarity Score
CBF	Content-Based Filtering
CF	Collaborative Filtering
CRS	Conversational Recommender System
FAISS	Facebook AI Similarity Search
GPT	Generative Pretrained Transformer
GPTScore	GPT-based Semantic Evaluation Score
HR@k	Hit Rate at rank k
LLM	Large Language Model
LLaMA	Large Language Model Meta AI
LoRA	Low-Rank Adaptation
MAE	Mean Absolute Error
nDCG	Normalized Discounted Cumulative Gain
NLI	Natural Language Inference
PEFT	Parameter-Efficient Fine-Tuning
QLoRA	Quantized Low-Rank Adaptation
RMSE	Root Mean Square Error
SBERT	Sentence-BERT
SHAP	SHapley Additive exPlanations
TMDB	The Movie Database
XGBoost	Extreme Gradient Boosting

Chapter 1

Introduction

Recommendation systems have become a key component of modern digital platforms. Whether suggesting a movie on a streaming service or selecting products in an online store, they help users navigate large collections of content by offering choices that align with their preferences. As they are adopted more broadly, the need grows not only for accurate predictions but also for explanations that users can understand and trust.

Collaborative and content-based filtering remain two of the most widely used approaches. These methods are used in many real-world systems and have been the focus of extensive academic and industry research. While they often deliver relevant suggestions, they typically do not provide any explanation for why a particular item appears among the recommendations.

Recent advances in language modeling have introduced new ways to enhance recommendation systems. Large language models are able to encode rich semantic information from text and generate coherent, natural language outputs. These capabilities make it possible to build systems that improve the quality of recommendations while also explaining their choices in natural language.

Rather than viewing large language models as replacements for traditional recommendation methods, it is worth exploring how they can be used in combination. Such integration can bring together the strengths of each approach and result in systems that are both accurate and more capable overall.

1.1 Scope of the Thesis

The goal of this thesis is to enhance recommendation systems through the use of large language models, with a focus on improving both predictive accuracy and explainability.

We begin by implementing a set of baseline recommendation models, including user-based and item-based collaborative filtering as well as a simple content-based approach. These are evaluated and compared against a proposed SBERT-based model, which we construct using enriched textual metadata, such as titles, genres, plot summaries, and director names, encoded into dense semantic embeddings. This semantic approach serves as an extension beyond traditional similarity methods, allowing for more meaningful item comparisons.

Having evaluated these models, we investigate whether their individual strengths can be combined to achieve better overall performance. Based on this idea, we develop a hybrid recommendation system that merges the outputs of the collaborative and SBERT-based approaches using a gradient boosting regressor. The hybrid is evaluated under the same experimental conditions as the previous models, covering both rating prediction and ranking performance, in order to enable consistent and meaningful comparisons.

With the core recommendation pipeline established, the thesis then turns to the question of explainability. Based on SBERT-generated recommendations, we build a dataset of structured prompts and natural language responses, where explanations are initially produced using GPT-3.5. These examples are used to fine-tune an LLaMA-3 model using parameter-efficient methods, QLoRA with LoRA adapters. The resulting model is able to generate fluent, grounded justifications that reference the user's preferences and the metadata of the recommended items. The explanations are evaluated using both automatic metrics, such as BERTScore, and structured GPT-based evaluation criteria, GPTScore-style, assessing their factual consistency, relevance, and clarity.

Finally, to move beyond static explanation, we explore how large language models can support interactive recommendations. We develop a conversational assistant that combines intent detection, semantic retrieval using SBERT and FAISS, and natural language responses driven by an LLM. This framework demonstrates how recommendations and explanations can be delivered through multi-turn dialogue.

1.1.1 Contributions

The main contributions of this thesis are:

- A semantic recommendation model based on SBERT, evaluated against traditional baselines in both rating prediction and ranking tasks.
- A hybrid architecture that combines collaborative and semantic signals through a gradient boosting regressor.
- A fine-tuned LLaMA-3 model capable of generating natural language explanations for recommendations, trained on GPT-3.5-generated examples using QLoRA.
- A structured evaluation framework for explanation quality, combining automatic metrics with GPT-based assessments.
- A conversational assistant that integrates recommendation, explanation, and user interaction through semantic retrieval and LLM-based dialogue.

1.2 Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 reviews related work on traditional, semantic, hybrid, and explainable recommendation models, as well as conversational systems. Chapter 3 describes the dataset and the enriched metadata representations used throughout the thesis. Chapter 4 introduces the baseline models and a proposed SBERT-based recommender. Chapter 5 describes the hybrid architecture that combines collaborative and semantic signals. Chapter 6 focuses on explainable recommendation through a fine-tuned LLaMA model and the evaluation of generated explanations. Chapter 7 presents a conversational assistant. Finally, Chapter 8 concludes the thesis and outlines directions for future work.

Chapter 2

Related Work

As mentioned earlier, this thesis focuses on three key areas: recommendation modeling, natural language explainability, and conversational interaction. Although each has been studied separately, advances in Large Language Models (LLMs) now make it possible to integrate them into more expressive and interactive systems. This chapter reviews relevant work in each area, beginning with recommendation methods, both classical and semantic, followed by explainability techniques using LLMs, and concluding with conversational recommender systems.

2.1 Recommendation Modeling

Traditional recommendation systems typically build on collaborative filtering (CF) and content-based filtering (CBF) to offer personalized suggestions. In CF, user preferences are inferred by comparing interaction patterns, either between users or between items. One of the earliest and most widely adopted item-based methods was introduced by Sarwar et al. [1], offering both scalability and acceptable performance in dense datasets. However, they often struggle in scenarios with sparse data or limited user interaction history. Both user-based and item-based CF serve as strong baselines and are also used as components in our later hybrid model.

Content-based filtering, on the other hand, makes recommendations by analyzing item attributes such as genres, keywords, or descriptive metadata. Although this method directly addresses the cold-start problem, it often fails to capture deeper user intent or suggest items outside of a user's narrow interest profile [2]. In our work, we use a simple content-based

model for comparison purposes.

To move beyond these limitations, recent work has turned to semantic modeling techniques. BERT4Rec [3], for example, demonstrates how bidirectional transformers can learn contextualized item representations from user interaction sequences. Inspired by these approaches, we adopt a similar strategy by embedding items into a semantic space using SBERT over enriched textual representations.

Rather than treating collaborative and semantic models separately, several approaches have explored hybrid architectures that combine their strengths. For example, Behera et al. [4] propose a hybrid recommendation model that integrates matrix factorization with additional item and user features, using XGBoost as a ranking model. Similarly, Singh et al. [5] combine collaborative filtering (via SVD), content-based similarity (TF-IDF), and boosting algorithms such as CatBoost and XGBoost to improve recommendation performance. Following these directions, we develop a hybrid model that brings together user-based and item-based CF signals along with SBERT-based semantic similarity, using XGBoost.

According to Burke's taxonomy of hybrid recommenders [6], our method falls under the metalevel category, where individual models act as components that feed into a final predictive learner. This setup allows greater flexibility in adapting to different input signals while preserving model's interpretability.

2.2 Explainability in Recommendation with LLMs

The ability to explain recommendations in natural language is becoming an increasingly important goal in recommender systems. Traditional approaches often rely on templates, hand-crafted rules, or attention-based signals to justify recommendations, but these methods tend to lack fluency and adaptability to user specific contexts.

Recent advances in large language models (LLMs) have enabled more expressive and flexible explanation generation. Ma et al. [7] propose XRec, a framework that augments a pre-trained LLM with collaborative filtering features to generate personalized justifications. Their approach focuses on grounding the output using interaction history and user-item embeddings. Similarly, Yang et al. [8] introduce LLM2ER, a fine-tuned LLM trained with a reinforcement learning objective that rewards high-quality explanations based on coherence and user alignment. Both works emphasize aligning explanations with user preferences. Our

method follows this direction by fine-tuning an open LLM (LLaMA 3 [9]) on explanation examples constructed from user history, using instructional prompts grounded in semantically similar items.

Evaluating the quality of generated explanations remains a key challenge. Zhang et al. [10] propose using large language models, such as GPT-4, as evaluators of recommendation explanations, showing that their judgments align well with human ratings in aspects such as relevance, transparency, and informativeness. In parallel, semantic similarity metrics such as BERTScore [11] and GPTScore [12] have been used to measure how closely generated explanations match reference outputs. Drawing on these insights, we evaluate our explanations using GPT-based models with structured prompts that reflect prior evaluation dimensions, and additionally compute BERTScore to assess semantic alignment between our fine-tuned LLaMA outputs and those generated by GPT-3.5.

2.3 Conversational Recommendation with LLMs

Conversational recommender systems are designed to support open-ended interaction, allowing users to explore options and gradually express their preferences. A challenge in conversational recommendation is combining multiple components, such as retrieval, classification, and generation, into a dialogue system that feels smooth and engaging. Zhou et al. [13] address this by introducing CRSLab, an open-source framework that connects these elements into an end-to-end conversational setup. Similarly, Lei et al. [14] propose an interaction cycle that includes estimation, action, and reflection, capturing the iterative nature of user-system interactions in dialogue-based recommendation.

With the rise of large language models, several studies have explored their role not just as generators, but as agents capable of reasoning and decision-making. Yao et al. [15] introduce ReAct prompting, allowing language models to both select an action and explain their reasoning. This approach has inspired new ways of using LLMs for candidate selection and justification in recommendation tasks. Hou et al. [16] further demonstrate that pretrained models can perform item ranking without explicit training, by using structured prompts that include candidate options and relevant user context.

Our approach builds on these ideas by combining semantic retrieval with LLM-based selection. The model chooses among candidate items and produces a natural-language justi-

fication, while also adapting to user feedback through simple dialogue state tracking.

Chapter 3

Dataset and Representations

In order to evaluate different recommendation strategies and train explanation models, we built our system on top of the widely used MovieLens 1M dataset [1], which provides a large-scale collection of user-movie ratings.

3.1 MovieLens 1M Dataset

The MovieLens 1M dataset consists of:

- 1,000,209 ratings from 6,040 users on 3,952 movies
- Ratings are on a scale from 1 to 5
- Each rating entry includes: `userId`, `movieId`, `rating`, and `timestamp`

The dataset provides basic movie metadata such as titles and genres, but lacks other descriptive elements that could support semantic or explainable recommendation approaches. Therefore, we enriched it with additional fields described in the next section.

3.2 Metadata Enrichment

To support semantic-based recommendation models, the original MovieLens 1M dataset was extended with additional metadata fields. These include textual and semantic information that allows for improved item representations and interactions based on natural language.

[1] `https://grouplens.org/datasets/movielens/1m/`

The enrichment process was performed using a combination of automated and manual techniques. Specifically, The Movie Database (TMDB) API was used to retrieve the following fields for each movie:

- **Title**: Original movie title
- **Genres**: A list of genres (e.g., *Action, Comedy*)
- **Director**: Name(s) of the film's director(s)
- **Overview**: A brief summary of the plot

To match MovieLens 1M movies with the corresponding TMDB entries, we used the `links.csv` file from the MovieLens 20M dataset, which provides mappings from the internal `movieId` values to the external TMDB and IMDb IDs. This allowed for accurate and efficient metadata retrieval in most cases.

Out of the total movies in the dataset, approximately 900 entries lacked a valid TMDB ID in the `links.csv` file. For these cases, a fallback strategy was applied:

- First, a search was performed using the movie title via the TMDB API.
- If no reliable match was found, a manual search was performed to verify and retrieve the correct metadata.

Additional normalization steps were applied to ensure consistency in matching:

- Lowercasing and trimming whitespace from titles
- Removing year suffixes or formatting tokens (e.g., `(2001)`)
- Applying fuzzy string matching in ambiguous cases

After enrichment, the metadata was merged with the original MovieLens entries to create a unified file. This file includes both the original and extended metadata and is used as the standard movie representation across all components of the system.

3.3 Textual Representation of Items

To enable semantic reasoning and natural language generation, we needed a way to represent each movie not just as an ID or a set of features, but as a structured textual description. This text field is used throughout the system, computing SBERT embeddings 4.3, building prompts for explanation generation 6.1, and retrieving results in the conversational assistant 7.3.

To achieve this, we designed a textual representation for every movie by combining its most important metadata into a single sentence style format. Each entry includes the title, genres, director, and a short overview of the plot. The template we used was:

```
Title: {title}. Genres: {genres}. Director: {director}.
Plot Summary: {overview}.
```

During this process, we also took care to handle missing values and formatting inconsistencies, ensuring that every movie has a clean and consistent description.

3.4 User Information

Each user in the MovieLens 1M dataset is identified by a unique `userId` and is associated with a set of explicit ratings for different movies. These ratings represent the user's expressed preference for each movie.

The dataset includes over 6,000 users, each with varying numbers of rated items. We do not apply any filtering or pruning based on user activity levels, and all users are retained as provided in the original dataset.

Our system focuses exclusively on item-based similarity and semantic modeling, without relying on additional user metadata such as age, gender, or occupation. Each user is instead represented implicitly through their rated movie history, which serves as the primary context for generating recommendations and explanations.

3.5 User-Item Rating Matrix

One of the key data structures used throughout our approaches is the user-item rating matrix, which encodes how users have rated different movies. We generate this matrix directly

from the `ratings.dat` file, which contains entries of the form `(userId, movieId, rating)`.

To represent the data efficiently, we map each unique user to a row index and each rated movie to a column index. These mappings are stored in Python dictionaries:

- `user_mapper`: maps each `userId` to a row index
- `movie_mapper`: maps each `movieId` to a column index
- Inverse mappings are also stored for reference

Using these indices, we create a sparse matrix $X \in \mathbb{R}^{M \times N}$ in `csr_matrix` format, where M is the number of users and N the number of movies that have received at least one rating. Each nonzero entry X_{ij} corresponds to the rating that user i gave to movie j.

Movies without any ratings are excluded entirely, as they are not represented in the ratings data and thus have no corresponding column in the matrix. This ensures that the matrix contains only items with observed user interactions.

Chapter 4

Recommendation Models

This part of the work focuses on different ways to generate recommendations, ranging from classic collaborative filtering techniques to models that rely on semantic similarity derived from textual metadata. Each method depends on a different source of information: how users interact with items, how items are rated, what content they contain, or how they are described textually.

We begin with collaborative filtering methods which use only the user-item rating matrix to identify patterns among users or items. Then, we examine a simple content-based method that compares items based on shared attributes, followed by a semantic model that represents items through sentence embeddings based on enriched metadata.

By evaluating these methods independently, we gain insight into their respective behaviors and limitations. This comparison motivates the development of a hybrid model that we will present in the next chapter.

4.1 Collaborative Filtering Approaches

Collaborative filtering is a widely used technique for recommendation tasks. It operates under the assumption that similar users tend to interact with similar items. In this work, we implement both user-based and item-based collaborative filtering using the user-item rating matrix constructed in Section 3.5.

4.1.1 User-Based Collaborative Filtering

In the user-based approach, the idea is that users who have rated items similarly in the past are likely to share preferences in the future as well.

Our implementation operates directly on the user-item matrix, where each row corresponds to a user and each column to a movie. Given the sparse nature of this matrix, not all users have rated the same items. To assess how similar two users are, we use cosine similarity between their respective rating vectors. For users u and v, similarity is computed as:

$$\text{sim}(u, v) = \frac{X_u \cdot X_v}{\|X_u\| \cdot \|X_v\|}$$

This similarity reflects how closely users agree in their ratings, based on the items they have both evaluated. For each target user, we identify the k users with the highest similarity scores. Their ratings are used to infer preferences for items that the target user has not rated.

The predicted rating for a movie is calculated as a weighted average of the neighbors' ratings for that movie, where the weights are their similarity to the target user. If none of the neighbors have rated the movie, no prediction is made. Otherwise, the predicted score reflects how positively the user's closest peers viewed the item.

This method does not rely on any item's features or textual metadata and instead depends entirely on rating patterns. As such, it performs well when users have overlapping histories, but tends to degrade when interactions are sparse or missing entirely.

4.1.2 Item-Based Collaborative Filtering

In contrast to the user-based method, the item-based approach focuses on the relationships between items. The idea is that a user is likely to enjoy items that resemble those they have rated positively in the past.

Rather than comparing users, we examine how items relate to each other based on shared rating patterns. Two movies are considered similar if they tend to receive comparable ratings from the same group of users. To capture this, we use the same cosine similarity measure introduced earlier but now apply it to the columns of the user-item matrix.

To generate a prediction, we look at the items the user has already rated, identify the ones most similar to the target item, and calculate a weighted average of the user's ratings for those similar items. This allows us to estimate how much the user might like a movie that

they haven't seen, based on their known preferences.

An advantage of this approach is that it tends to be more stable in environments with many users, since item similarities change less frequently than user profiles. Still, it faces similar limitations to the user-based approach, especially in cases where little feedback exists for new items.

4.2 Content-Based Filtering

As a baseline model, we implemented a simple content-based filtering approach that recommends movies based on genre similarity. The underlying assumption is that users tend to enjoy movies that belong to the same genres as the ones they have already liked.

Each movie is encoded as a binary vector that marks which genres are associated with it. Using these vectors, we calculate cosine similarity between movies, resulting in an item-item matrix where each entry reflects how closely two movies align in terms of genre.

To generate predictions, we consider the movies that the user has already rated. For each unseen movie, we select the k most similar items the user has already rated and estimate a weighted average of those ratings using genre similarity as weights.

Although this approach is easy to implement and requires minimal data, it has clear limitations. It assumes that all genres matter equally and does not account for combinations or patterns that might better explain a user's interests.

4.3 Semantic Model (SBERT-Based)

In recommendation systems, structured features such as genres or ratings can only capture part of what defines a movie. Often, plot, themes, and style hold information that is difficult to express using categories alone. In this section, we describe an alternative approach that focuses on the written descriptions of movies, aiming to compare them in a way that reflects their overall meaning and content.

4.3.1 From BERT to SBERT: Learning Semantic Representations

Our method is based on **Sentence-BERT (SBERT)** [17], a model built on top of **BERT** [18], which is one of the most influential models in natural language processing. Although BERT

was originally designed for general purpose language understanding, it does not perform well for comparing full sentences or texts directly. SBERT improves on this by using a special training setup that teaches the model to recognize how similar two sentences are in meaning. This is achieved by pairing or grouping sentences during training and teaching the model to position those with similar meanings closer together in the embedding space.

In practice, SBERT turns each input text, such as a movie description, into a fixed size representation by averaging the output of the model across all tokens. These vectors are designed so that similar texts end up close to each other when compared using cosine similarity. Because of this, SBERT has become a common choice for tasks that involve comparing and ranking text, especially in information retrieval and semantic search settings [19, 20].

4.3.2 Textual Representation of Movies

To use SBERT in our system, we first built a short text for each movie that brings together its title, genres, director, and a brief description of the plot. As explained in Section 3.3, these elements were joined into a single text field. This description was used as input to the SBERT model to produce a fixed-size embedding for each movie.

4.3.3 Model Selection: Why MPNet over MiniLM

We explored multiple SBERT variants available through the Sentence-Transformers library [17]. Our initial experiments used `all-MiniLM-L6-v2`, which is a fast and lightweight model. However, we noticed that `all-mpnet-base-v2` produced better and more consistent results in our ranking tasks. This observation matches findings from prior work [21], which show that MPNet-based models tend to perform better on tasks involving sentence similarity, thanks to improvements in how they capture word order and context [22].

4.3.4 Recommendation Strategy Based on SBERT Similarity

To generate recommendations, we compare each unseen movie with all movies that the user has already watched, based on the cosine similarity of their SBERT embeddings. Instead of representing the user with a single averaged vector, we follow an item-item strategy, where the relevance of a candidate movie is estimated from its similarity to the user's previously seen movies.

For each movie j that the user has not rated, we calculate its cosine similarity with all movies $i \in I_u$, where I_u is the set of movies the user has rated. The final score is then computed as a weighted average, where each similarity is weighted by the user's rating for the corresponding movie:

$$\text{score}(u, j) = \frac{\sum_{i \in I_u} r_{ui} \cdot \cos(\vec{m}_j, \vec{m}_i)}{\sum_{i \in I_u} r_{ui}}$$

Here, r_{ui} is the rating that user u gave to movie i, and $\vec{m}_j$, $\vec{m}_i$ are the SBERT embeddings of the candidate and watched movies respectively.

This method preserves individual preferences by giving more importance to movies the user rated highly. In our evaluation, it consistently outperformed user-level averaging approaches.

4.3.5 Strengths and Considerations

A key benefit of this method is that it allows the system to recognize connections between movies based on their story, themes, or tone, even if these connections are not reflected in structured metadata. This is especially helpful when dealing with new or rarely rated items, where collaborative methods have little information to rely on. Another advantage is that these embeddings are also useful in other parts of the system, such as generating explanations or searching through movies in a dialogue.

However, there are some things to keep in mind. The model can only be as good as the descriptions it receives. If a movie's description is poorly written or lacks important details, the resulting embedding may not capture what the film is actually about. Additionally, although MPNet is more efficient than larger transformer models, generating embeddings for thousands of movies still requires a significant amount of time and memory.

4.4 Evaluation of Baseline and Semantic Models

4.4.1 Evaluation Protocols

Recommendation systems can be evaluated in multiple ways, depending on the nature of the task and the type of output the model produces. When the system predicts explicit ratings, it makes sense to assess how close its predictions are to the actual user-provided values. On

the other hand, when the goal is to recommend a list of items, what matters is how well the system ranks relevant items among the top suggestions.

In our work, we adopt both strategies. The first focuses on rating prediction and evaluates the numerical accuracy of the model. The second focuses on ranked retrieval and measures whether the right items appear in the top positions. This distinction is consistent with what is suggested in the literature. For example, Gunawardana and Shani [23] emphasize that evaluation protocols should match the application scenario. Similarly, Herlocker et al. [24] highlight the importance of aligning evaluation metrics with the system's intended use, distinguishing between rating prediction and ranking-oriented tasks.

4.4.2 Rating Prediction Evaluation

In this evaluation setup, the goal is to assess how accurately each model can predict the actual ratings given by users. We apply this strategy to all models that produce numerical predictions, including collaborative filtering, content-based, and SBERT-based approaches. The dataset is split into training and test sets using a standard 80/20 ratio. For each user–item pair in the test set, we compare the predicted rating to the true rating provided by the user.

We report three common error metrics:

- **Mean Absolute Error (MAE)**:

$$\text{MAE} = \frac{1}{N} \sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}|$$

  where $\hat{r}_{ui}$ is the predicted rating, r_{ui} is the true rating, and $\mathcal{T}$ is the set of test instances.

- **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{N} \sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2$$

- **Root Mean Squared Error (RMSE)**:

$$\text{RMSE} = \sqrt{\text{MSE}}$$

  RMSE penalizes larger errors more heavily than MAE and is commonly used in rating-based evaluations.

In addition to these error-based metrics, we also report the model's **Coverage**, which measures the proportion of test instances for which the model is able to return a prediction:

$$\text{Coverage} = \frac{|\{(u,i) \in T : \hat{r}_{ui} \text{ is defined}\}|}{|T|} \times 100\%$$

where T denotes the test set and $\hat{r}_{ui}$ is the predicted rating for user u and item i.

Coverage is particularly important in sparse datasets like MovieLens 1M, where traditional collaborative filtering methods may not be able to return predictions for all user–item pairs. Low Coverage indicates that the model fails to generalize to many evaluation cases, regardless of its accuracy, where predictions are possible.

This form of evaluation is commonly used for models that output explicit rating values [24] and provides a straightforward measure of predictive accuracy across different methods.

4.4.3 Ranking Evaluation

To assess how well a model produces ranked recommendation lists, we adopt a ranking-based protocol inspired by recent work on top-N recommendations. For each user, we select one held-out item that they have rated positively and sample 99 items with which they have not interacted. The model is then asked to rank all 100 candidates, ideally placing the relevant item near the top.

We compute the following metrics:

- **Hit@K**: Measures whether the positive item appears in the top K predictions. It is defined as:

$$\text{Hit@K} = \begin{cases} 1 & \text{if the relevant item is ranked in the top } K \\ 0 & \text{otherwise} \end{cases}$$

  This metric is averaged over all users to obtain the final score. It provides a simple binary signal of success: did the model manage to include the correct item among its top suggestions? While it ignores rank ordering within the top K, it remains useful in cases where evaluation focuses solely on whether a relevant item was retrieved.

- **Normalized Discounted Cumulative Gain (NDCG@K)**: Accounts for the position of the relevant item in the ranked list, assigning higher credit when it appears earlier. For a single instance, it is calculated as:

$$\text{NDCG@K} = \frac{1}{\log_2(r+1)} \quad \text{(since IDCG = 1)}$$

where r is the rank position of the relevant item (1-based indexing). If the item is not in the top K, the NDCG score is set to 0. In our setting, each test case includes a single relevant item, so the ideal DCG is always 1. This allows us to compute NDCG directly using a simplified formula without explicitly calculating DCG or performing normalization.

This setup reflects realistic use cases where only a handful of suggestions are shown to the user. It also provides a more fine-grained evaluation of ranking quality. The approach follows the methodology proposed by He et al. [25], who use leave-one-out evaluation combined with fixed-size negative sampling to test top-N recommender models efficiently.

4.4.4 Fallbacks and Sparsity Issues

Collaborative filtering methods rely heavily on the availability of shared user–item interactions. In sparse datasets like MovieLens 1M, it is common for a user's neighbors to have no ratings for many of the items tested, particularly when those items were randomly sampled. As a result, models such as user-based and item-based CF may not produce valid predictions for a significant number of candidate items.

When this happens, systems typically fall back to default values, such as the user's average rating or a constant baseline. Although this ensures that a score is given, it does not reflect any pattern learned from similar users or items.

We found that both collaborative filtering models frequently ended up using fallback values when there was not enough data from similar users or items. To understand the scale of the issue, we measured how many of the predictions for each user were based on fallback rather than real ratings. This offered a clearer view of how much sparsity affected the results, something that is not always visible through ranking scores alone. As noted by Grcar et al. [26], such predictions should be treated carefully, as they do not reflect the actual logic of the model.

In contrast, our SBERT-based model does not suffer from this issue. Since it computes similarity using text embeddings rather than rating overlap, it produces ranking scores for all candidates, resulting in full Coverage during the evaluation.

4.4.5 Rating Prediction Results

In this section, we evaluate the predictive performance of the four models, user-based CF, item-based CF, content-based filtering, and SBERT, using the error metrics MAE, RMSE along with Coverage introduced earlier, in Subsection 4.4.2.

We report two sets of results. We first analyze how performance varies with the number of neighbors k, using a fixed 80/20 train–test split. Then we examine how the test split ratio affects model accuracy, using $k = 20$ for all methods. This value was selected as a representative trade-off between accuracy and coverage.

Tables 4.1 and 4.2 present the full results across values of k. Tables 4.3 and 4.4 report under different test sizes. The best scores per column are **in bold**.

Table 4.1: MAE and RMSE for each model across different neighborhood sizes k.

Model	**MAE (k)**					**RMSE (k)**				
	10	**20**	**30**	**40**	**50**	**10**	**20**	**30**	**40**	**50**
User-Based CF	0.827	0.801	0.789	0.783	0.779	1.068	1.022	1.002	0.993	0.986
Item-Based CF	**0.735**	**0.724**	**0.722**	**0.722**	**0.723**	**0.979**	**0.948**	**0.939**	**0.935**	**0.935**
Content-Based	0.827	0.815	0.812	0.811	0.811	1.049	1.031	1.026	1.025	1.024
SBERT	0.799	0.791	0.792	0.793	0.796	1.009	0.996	0.994	0.996	0.998

Table 4.2: Coverage (%) of each model across different neighborhood sizes k.

Model	**Coverage (k)**				
	10	**20**	**30**	**40**	**50**
User-Based CF	88.8	94.3	96.2	97.3	97.9
Item-Based CF	91.4	95.8	97.3	98.1	98.5
Content-Based	99.8	99.8	99.8	99.8	99.8
SBERT	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 4.3: MAE and RMSE for each model across different test set sizes ($k = 20$).

Model	MAE (Test Split)				RMSE (Test Split)			
	10%	**20%**	**30%**	**40%**	**10%**	**20%**	**30%**	**40%**
User-Based CF	0.793	0.801	0.810	0.822	1.010	1.022	1.036	1.056
Item-Based CF	**0.719**	**0.724**	**0.731**	**0.738**	**0.940**	**0.948**	**0.960**	**0.972**
Content-Based	0.812	0.815	0.816	0.819	1.028	1.031	1.034	1.039
SBERT	0.790	0.791	0.793	0.796	0.994	0.996	0.998	1.001

Table 4.4: Coverage (%) for each model across different test set sizes ($k = 20$).

Model	Coverage (Test Split)			
	10%	**20%**	**30%**	**40%**
User-Based CF	95.2	94.3	93.0	91.0
Item-Based CF	96.4	95.8	94.8	93.3
Content-Based	99.8	99.8	99.8	99.7
SBERT	**100.0**	**100.0**	**100.0**	**100.0**

Across different values of k, item-based CF consistently provides the most accurate predictions, especially as the neighborhood grows. SBERT comes close in terms of error, while offering the clear advantage of full Coverage regardless of setup. User-based CF improves with larger k, but remains more sensitive to missing data. The content-based model shows weaker predictive accuracy overall, although its Coverage stays high across the board.

When the test split increases, all models show the expected decline in both accuracy and Coverage, especially the collaborative models, which depend on sufficient overlap between users or items. SBERT, by contrast, stays remarkably stable. Its error metrics vary only slightly and its Coverage remains perfect even when less training data is available. This kind of consistency makes it a reliable option, especially in sparse data.

4.4.6 Ranking Evaluation Results

In this section, we present the results of the ranking evaluation, based on the protocol introduced earlier. For each user, we evaluate the model's ability to rank one held-out posi-

tive item among 99 randomly selected negatives. We report Hit@10 and NDCG@10, which capture how well the relevant item appears near the top of the recommendation list.

Alongside ranking performance, we also track the average Fallback ratio per user, how often a model returns a default score instead of a prediction.

All results are reported with $k = 20$. This value was selected as a representative neighborhood size that balances relevance and Coverage in the context of collaborative filtering. Although higher values of k can increase the chance of finding overlapping neighbors, thus reducing Fallback rate, they often incorporate weaker or less informative neighbors, which can negatively affect ranking precision. Using $k = 20$ offers a realistic trade-off under the sparsity typical of real-world data.

Table 4.5: Ranking performance and Fallback ratios ($k = 20$).

Model	**Hit@10**	**NDCG@10**	**Fallback (%)**
User-CF	0.5650	0.3046	74.81
Item-CF	0.5480	0.2663	73.96
CBF	0.1380	0.0699	7.17
SBERT	0.4390	0.2494	**0.00**

User-based CF achieved the highest Hit@10, but at the cost of a high Fallback rate: nearly 75% of its predictions defaulted to a fallback value due to missing neighbor overlap, undermining the reliability of its rankings. Item-based CF exhibited a similar pattern, with slightly lower ranking scores and an almost identical Fallback ratio.

Despite achieving full Coverage, the content-based model performed poorly in the ranking task. It fails to distinguish truly relevant items from unrelated ones, particularly when ranking among a large set of candidates.

The SBERT-based model, while not achieving the highest Hit@10, yielded competitive NDCG@10 scores and was the only method to provide valid predictions for all user–item pairs. Its zero Fallback rate, coupled with solid ranking performance, underscores its reliability in sparse data scenarios. Unlike neighborhood-based methods, its embedding-based approach allows it to retrieve relevant candidates even when user history lacks overlap with the test items.

4.5 Summary and Transition

The results of the rating and the ranking evaluation show that the SBERT-based model performs more reliably than traditional approaches, particularly in sparse settings. Even though it does not always reach the lowest prediction error, it avoids fallback behavior entirely and produces meaningful recommendations for all users. These findings motivate the use of SBERT as a core component in more advanced systems. In the next chapter, we explore how these individual models can be combined into a hybrid approach that brings their strengths into a single model.

Chapter 5

Hybrid Model

As our previous evaluation demonstrated, each recommendation method has its own strengths and limitations. Techniques based on collaborative filtering tend to perform well when there is plenty of rating data, but their effectiveness drops sharply for users with fewer interactions. Content-based models take a different approach, focusing on item attributes like genre or description. These are often more stable in sparse settings, yet they usually struggle to reflect user preferences in a meaningful way.

The SBERT-based model offered a reliable alternative. It handled sparsity gracefully and avoided fallback behavior, producing coherent suggestions even when little data was available. Still, in terms of raw prediction accuracy, traditional methods, particularly item-based collaborative filtering, often had the upper hand.

Rather than selecting a single method, we investigate whether different models can be combined so that each contributes when it is most effective. The idea is to use collaborative signals when they are strong, and fall back on semantic similarity when interaction data is limited. This approach aims to improve accuracy while ensuring that recommendations remain available for all users.

Among the possible strategies for bringing models together, model-based combinations have proven especially effective [6, 4]. In this setup, a separate learner figures out how to make use of the predictions from each component. We adopt this strategy by training an XGBoost regressor to integrate the outputs of three models, item-based CF, user-based CF, and a semantic SBERT-based method, into a single prediction that captures their complementary strengths.

5.1 Model Architecture

The hybrid model follows a stacked architecture, combining predictions from three independent recommenders: item-based CF, user-based CF, and a semantic model based on SBERT embeddings. By including these three components, the ensemble captures similarity between users, relations between items, and semantic information derived from textual content.

A genre-based content model was excluded, as its predictions overlapped with those of SBERT and did not lead to any noticeable improvement.

Each of the three models receives the same input, a user and a target movie, and returns a numerical score indicating its predicted rating. These predictions were all generated using a consistent neighborhood size of $k = 20$. The resulting scores are then assembled into a feature vector, which serves as input to a higher-level model. This final stage is implemented using an XGBoost regressor, trained to learn how to weigh and combine the base predictions. The regressor produces the final output, which corresponds to the estimated rating for that specific user–movie combination.

Figure 5.1 illustrates the overall structure of the hybrid model.

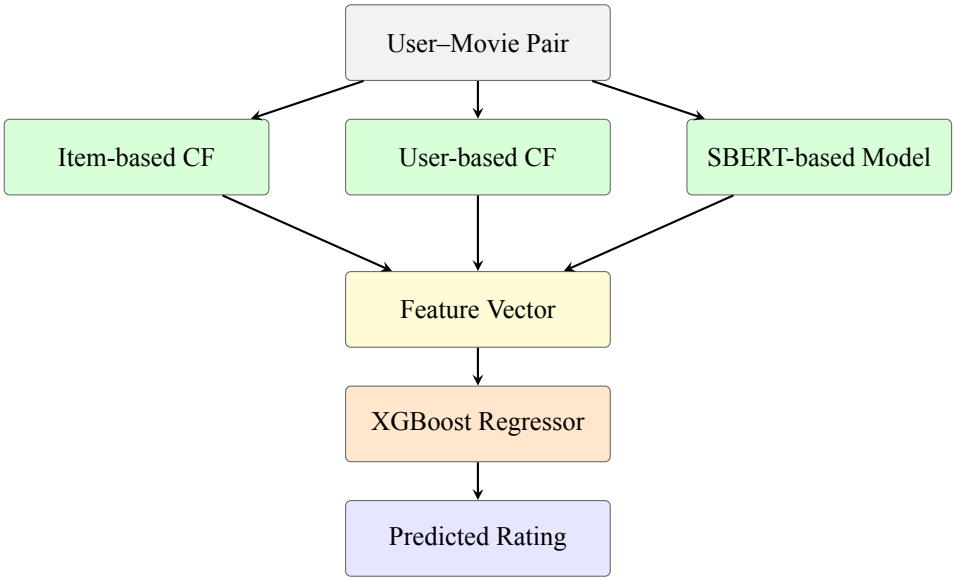

Figure 5.1: Stacking architecture of the hybrid model, combining predictions from three base recommenders.

5.2 Feature Normalization and Fallback Handling

All three individual models output scores on the same 0–5 scale, so no further normalization is required. However, collaborative filtering models may fail to produce a prediction when no suitable neighbors are found, typically due to data sparsity. In such cases, we assign a score of zero to indicate the absence of a prediction, which implicitly reflects a fallback behavior in the corresponding base model. This allows the hybrid model to distinguish between low-confidence and high-confidence signals without introducing bias.

The SBERT-based model, by contrast, is designed to provide output for any user–movie pair. This ensures that the final hybrid system always receives at least one valid input, effectively avoiding cold-start issues and maintaining full Coverage.

5.3 Training Protocol

To train and evaluate the hybrid model, we split the dataset into three separate subsets. The first 60% was used to train the individual base models. The next 20% served as the input to the hybrid model, where we generated training examples by pairing predicted scores with actual user ratings. The final 20% was held out as a test set, used for evaluating both the base models and the hybrid. This clear separation helped avoid any data leakage between stages and ensured a fair comparison across all models.

Table 5.1: Three-way split of the dataset used to train and evaluate the hybrid model.

Subset	**Portion**	**Used For**
Training Set	60%	Training the base models (CF, SBERT)
Validation Set	20%	Generating training data for the hybrid model
Test Set	20%	Final evaluation of all models

We used squared error as the loss function (`reg:squarederror`) and trained the model with 100 boosting rounds, a maximum depth of 4, and a learning rate of 0.1. The selected hyperparameters were established through exploratory experimentation and demonstrated consistently strong performance on the validation set.

We decided to use a gradient-boosted model rather than a linear regressor or a neural

network, mainly because it offers a good balance between flexibility, efficiency, and transparency. XGBoost can capture non-linear interactions between features without requiring careful normalization or extensive parameter tuning, which made it a practical choice for our stacked setup.

5.4 Evaluation and Model Insights

5.4.1 Evaluation Setup

To evaluate the hybrid model, we used the same protocols established in Subsection 4.4.1. For rating prediction, we measured the mean absolute error, the root mean squared error, and the Coverage across the test set. For ranking performance, we adopted a fixed negative sampling strategy and reported Hit@5, Hit@10, NDCG@5, and NDGG@10 scores.

It should be noted that the collaborative filtering models were not included in the ranking evaluation. When trained on only 60% of the data, these models failed to produce valid predictions for a significant portion of users, due to lack of overlapping ratings. In those cases, the system fell back to default outputs. A detailed analysis of this issue appears in Subsection 4.4.4, where we discuss the limitations of CF-based models under sparse conditions.

By contrast, the hybrid and SBERT-based models maintained full Coverage, allowing for reliable evaluation across all users.

5.4.2 Rating Prediction Results

Table 5.2 shows the rating prediction performance of each model on the test set. The hybrid model achieved the lowest error across all metrics, outperforming both collaborative filtering and SBERT-based approaches in terms of MAE and RMSE. The item-based CF model remained competitive, particularly in MAE, but its Coverage was slightly below full due to its reliance on neighborhood information. As expected, the user-based model performed the weakest, both in accuracy and Coverage.

Notably, both the SBERT and hybrid models achieved complete Coverage, meaning they produced predictions for all user–movie pairs. This consistency, combined with strong accuracy, demonstrates the benefit of using a stacked model that integrates different sources of information.

Table 5.2: Rating prediction performance of all models on the test set.

Model	RMSE	MAE	Coverage (%)
Item-based CF	0.9712	0.7373	93.4
User-based CF	1.0557	0.8232	91.0
SBERT	1.0003	0.7960	**100.0**
Hybrid (XGBoost)	**0.9033**	**0.7136**	**100.0**

5.4.3 Ranking Evaluation Results

We evaluated the ranking performance of the hybrid and SBERT-based models using Hit Rate and NDCG at both cutoff values $k = 5$ and $k = 10$, under a fixed negative sampling protocol. The results are shown in Table 5.3.

Table 5.3: Ranking performance for SBERT and Hybrid models.

Model	Hit@5	NDCG@5	Hit@10	NDCG@10
SBERT	0.3140	0.2177	0.4640	0.2771
Hybrid (XGBoost)	**0.4790**	**0.3454**	**0.6010**	**0.3852**

The hybrid model outperforms SBERT in all four metrics. At the standard cutoff of $k = 10$, it improves Hit Rate by over 13 percentage points and NDCG by more than 10. The gap widens further at $k = 5$, where the hybrid model maintains stronger top-ranked accuracy, suggesting it places relevant items higher in the list. Since these are the two most reliable models in our setup, evaluating both under stricter thresholds helps reveal clearer differences in ranking quality.

5.4.4 Coverage and Contribution Patterns

To analyze how the hybrid model utilizes its inputs, we examined the Coverage of each base recommender on the test set. For every user–movie pair, we checked which of the three models, item-based CF, user-based CF, and SBERT, returned a non-zero score, indicating a valid prediction. This allowed us to quantify the relative contribution of each model to the hybrid's input features.

The results are shown in Figure 5.2. In almost 88% of the test cases, all three models provided predictions. This indicates that the hybrid system was not just falling back on one or two signals, but actively combining all three in the majority of cases. These findings confirm that it operates within a true multi-model architecture.

In the remaining cases, SBERT consistently produced a score whenever the other models failed to do so. Whether combined with item-based CF, user-based CF, or used alone, it effectively filled the gap, which is expected given that it was the only model with full Coverage.

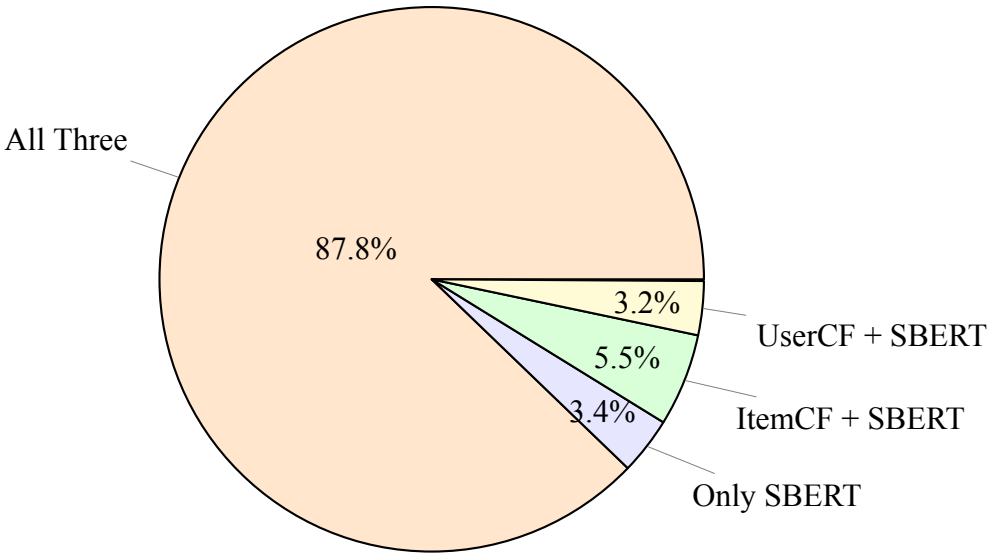

Figure 5.2: Distribution of base model combinations contributing to the hybrid model's predictions on the test set.

5.4.5 SHAP Analysis

To understand how the hybrid model combines its inputs, we used SHAP [27], a method designed to explain machine learning predictions by breaking them down into individual feature contributions. Inspired by concepts from game theory, SHAP assigns each input a value that reflects how much it influenced the final output, positively or negatively.

One of its main advantages is that it reveals patterns across the entire dataset while also explaining the reasoning behind specific predictions. This is especially useful in models like ours, where the weight of each component can vary from case to case.

Figure 5.3 shows how the three input signals shaped predictions in the test set. Each point corresponds to a user–movie pair. The position on the horizontal axis indicates the impact on

the predicted rating, while the color shows how large the input value was, blue for low, red for high.

Among the three, the item-based component had the clearest and strongest influence. High values almost always pushed predictions upward. The SBERT-based signal also played a key role, especially when collaborative methods lacked sufficient data. The user-based input had a smaller effect overall, which aligns with its lower Coverage and weaker performance.

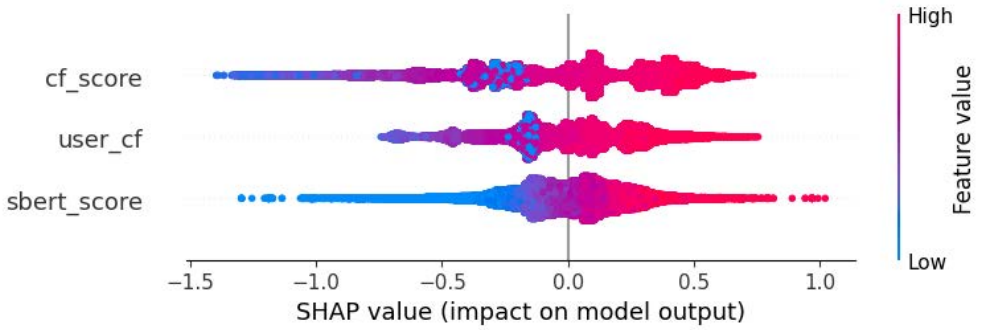

Figure 5.3: SHAP summary plot.

To make this more specific, we examined a single prediction in detail. In the example of Figure 5.4, the model starts from its baseline, the average rating, and adjusts it according to the available signals. The item-based value increases the score by 0.48, the SBERT input adds another 0.27, and the user-based signal brings the result down. The final prediction of 3.88 reflects how these signals combined.

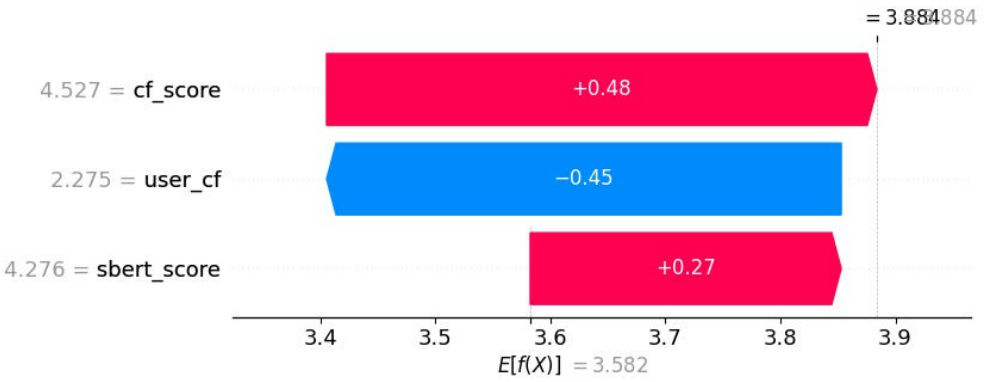

Figure 5.4: SHAP waterfall plot for a single prediction.

This breakdown shows the model's adaptive behavior that does not treat all signals equally, but shifts its focus depending on which ones are present and how much they help. SHAP makes this reasoning visible.

5.5 Summary and Transition

This chapter presented the design and evaluation of a hybrid recommendation model that combines three distinct approaches, item-based CF, user-based CF, and SBERT-based. We showed how a simple stacking model can learn to trust the right signals in the right context, improving both rating prediction and ranking performance over individual models.

In particular, the hybrid system outperformed SBERT in all ranking metrics while also retaining full Coverage, something that collaborative methods alone could not achieve. Model contribution and SHAP analysis further confirmed that the hybrid model dynamically adjusts its behavior based on available information, favoring collaborative inputs when they are strong and falling back to SBERT when needed.

In the next chapter, we shift our focus to a different kind of enhancement: instead of boosting accuracy, we explore how to generate natural language explanations for recommendations, making the system more transparent, interactive, and trustworthy.

Chapter 6

Explainable Recommendations

Our goal in this work is to move beyond black-box recommendations by generating natural language explanations that are grounded in the user's past preferences. Instead of simply saying *"You might like this movie"*, we want the system to say something closer to *"We recommend this because it shares themes with films you have rated highly and it is directed by someone whose work you tend to enjoy."*

To achieve this, we explored a language-model-driven approach by fine-tuning a pre-trained LLaMA 3 model [9] on explanations originally generated by GPT-3.5. This strategy allows us to combine the expressive power of large language models with the personalization of recommender systems. By training the model on examples that link user history with recommended items, we aim to teach it how to produce fluent, context-aware justifications that make intuitive sense to users.

In the sections that follow, we describe how this dataset of explanations was constructed, how the model was fine-tuned, and how we evaluated the resulting system.

6.1 Dataset Construction

6.1.1 SBERT-Based Recommendations

The first step in building our explanation dataset was to generate actual recommendations for users. To do this, we used our SBERT-based recommendation model. We chose SBERT over collaborative methods not only because it performed best in our evaluations, but also because its content-based nature makes its suggestions more interpretable. Since recommendations are based on semantic similarity in textual metadata, title, genres, director,

and overview, they can be more naturally explained in language, without relying on behaviors of other users.

6.1.2 Identifying Supporting Movies from User History

Once a recommendation was selected for a given user, we searched their viewing history to find which of their previously rated movies were most similar to the one being recommended. For each recommended movie, we retrieved the top-3 most similar movies the user had already seen, based on cosine similarity in the SBERT embedding space.

These similar, previously rated movies were used to support the explanation. Because they were part of the user's actual history and had ratings, they made it possible to refer to specific elements the user seemed to care about, like genre, storyline, or director.

6.1.3 Constructing the Explanation Prompts

With the recommendations and supporting movies already selected, the next step was to turn this information into prompts that a language model could understand and respond to naturally. Each prompt included the basic details of the recommended movie, along with metadata for three similar movies the user had previously rated. For every movie, we included the title, genres, director, a brief summary, the user's rating, and its similarity to the recommendation.

We then asked the model to justify the suggestion in exactly two sentences. The prompt made it clear that the explanation should begin with the phrase *"We recommend this movie because..."* and should refer to at least two of the previously seen movies. The goal was to encourage grounded, user-specific reasoning, highlighting shared themes, genres, or directors, without drifting into vague or generic statements.

Below is a shortened version of the prompt format we used. A complete example with all fields is provided in Appendix A.

You are a friendly and knowledgeable movie recommendation assistant.

Here is what the user has previously watched and rated:

- **Home Alone (1990)**
  Genres: Children's|Comedy, Director: Chris Columbus, Rating: 2, Similarity: 0.4817
  Summary: Eight-year-old Kevin McCallister makes the most of the situation...
- **Welcome to the Dollhouse (1995)**
  Genres: Comedy|Drama, Director: Todd Solondz, Rating: 5, Similarity: 0.4815
  Summary: An unattractive 7th grader struggles to cope with suburban life as the middle child...
- **Mallrats (1995)**
  Genres: Comedy, Director: Kevin Smith, Rating: 4, Similarity: 0.4801
  Summary: Both dumped by their girlfriends, two best friends seek...

Now you are recommending the following movie:

- **North (1994)**
  Genres: Comedy, Director: Rob Reiner
  Summary: Eleven-year-old North has had it with his parents...

Write exactly two sentences that explain to the user why this movie is recommended. Start with: *"We recommend this movie because..."*.

6.1.4 Generating Explanations with GPT-3.5

We applied this process to 1,000 users, generating a corresponding set of prompts like the one shown above. Each was submitted to GPT-3.5, which followed the instructions and returned a two-sentence explanation grounded in the user's previous preferences.

An example of a generated explanation is shown below:

> *We recommend this movie because "North" shares a similar genre of comedy with "Welcome to the Dollhouse" and "Mallrats", which you rated highly. Additionally, like "Home Alone", "North" also revolves around a young protagonist navigating family dynamics in a humorous and heartwarming way.*

6.2 Model Fine-Tuning

Having collected a dataset of 1,000 personalized explanations, the next step was to teach a language model to generate such outputs on its own. The goal was simple: replace GPT-3.5

with a smaller, open model that could produce equally natural and context-aware justifications, but without relying on external APIs or cloud access.

We, therefore, fine-tuned the `Meta-Llama-3-8B-Instruct` model on our prompt–completion pairs. We used parameter efficient training techniques to make the process more manageable, even with limited hardware. Specifically, we applied QLoRA [28], a quantization-aware method that reduces memory usage, alongside LoRA adapters [29] that fine-tune only a subset of the model's parameters.

The entire setup was built using the Hugging Face `transformers` [30], `peft` [31] and `trl` [32] libraries. The final model is fully local, lightweight enough to run on a single high-memory GPU, and capable of producing rich, personalized explanations based on user history and recommendation metadata.

6.2.1 Preparing the Tokenizer and Dataset

We began by loading the tokenizer for the base model, `meta-llama/Meta-Llama-3-8B-Instruct`. Since LLaMA 3 does not define a dedicated padding token, we assigned the end-of-sequence marker to serve that role as well. This is required for stable training when using FP16 precision.

Our dataset consisted of 1,000 examples in JSONL format, with each line containing a complete prompt response pair. Every entry followed the LLaMA 3 instruct-style format: a single string with both the prompt and the expected output, wrapped in `<|begin_of_text|>` and `<|end_of_text|>` tokens. We set the input length limit to 2048 to make sure both the context and response could fit comfortably in a single sequence.

6.2.2 Training Setup

To make training feasible on a single GPU, we loaded the base model using 4-bit weights, following the QLoRA methodology. This configuration significantly reduced the memory footprint while preserving the model's core capabilities. All computations were performed in half-precision (`float16`) and we enabled double quantization to further optimize resource usage.

All training was conducted on a `NVIDIA A40 GPU` with 48GB of VRAM, which provided sufficient memory to support this setup.

We then applied LoRA adapters to a small number of attention layers, allowing the model to adapt while keeping most weights frozen. Specifically, we targeted the query, key, value, and output projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`) with a low-rank configuration (`r=16`, `alpha=32`) and a dropout rate of 5%.

The model was trained for three epochs with an effective batch size of 4, accumulated over two gradient steps. We used a learning rate of `2e-4` and saved checkpoints at the end of each epoch. The final checkpoint was stored locally and serves as the explainable recommendation model we use throughout the rest of this work. We monitored training to ensure everything ran smoothly, but since our goal was not to optimize loss but to produce useful explanations, we did not use loss values for early stopping or model selection.

6.2.3 Inference and Model Use

Once fine-tuned, the model is ready to generate explanations for new recommendations. At inference time, we supply it with the same type of structured prompt used during training, a recommended movie along with metadata for a few similar titles the user has previously rated.

The model reads this input as a single block of text and produces a natural language explanation in two sentences, beginning with the phrase *"We recommend this movie because..."* and drawing explicit links to at least two supporting movies. This generation step is fast, runs entirely offline, and can be integrated into any recommendation interface to provide users with clear and personalized justifications.

To summarize the overall process, Figure 6.1 illustrates the full pipeline we used to build the explainable recommendation model.

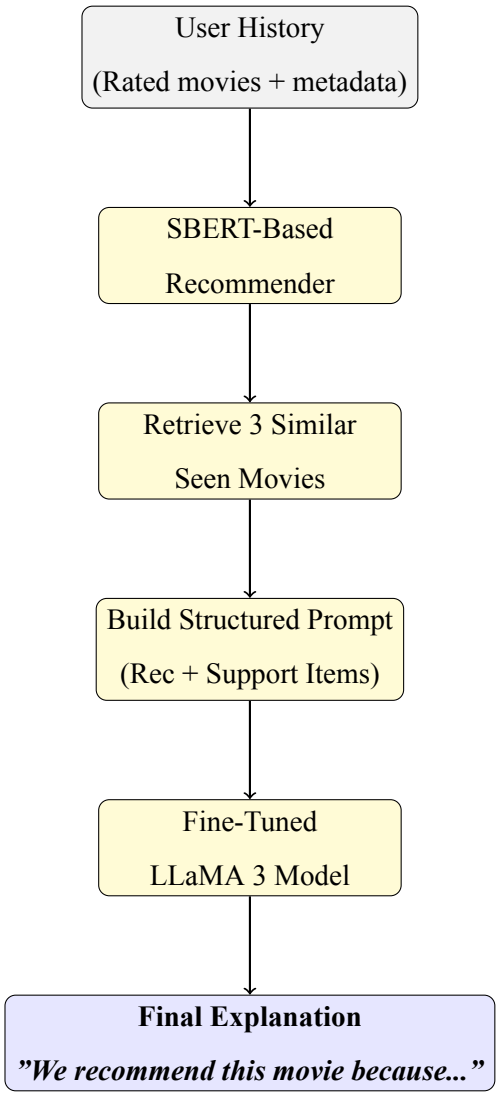

Figure 6.1: End-to-end pipeline for generating natural language explanations.

6.3 Evaluation of Generated Explanations

Evaluating natural language explanations is not as straightforward as measuring accuracy or precision in classic recommendation tasks. There is no single metric that captures what makes an explanation truly helpful or well-formed and the research community has not yet settled on a standard approach. As we discussed earlier in Section 2.2, recent work has explored several directions, from semantic similarity scores like BERTScore [11], to asking large language models such as GPT-4 to act as evaluators [10, 12].

In our case, we combined insights from these directions to design a small set of evaluation strategies that reflect different aspects of quality. Rather than relying on one metric, we wanted to understand how well our model performs across several fronts: how close its outputs are to those of GPT-3.5, whether they are factually consistent and structurally correct, and how helpful or personalized they feel.

The rest of this section presents the results of these evaluations, with both scores and observations.

6.3.1 BERTScore

To get an initial sense of how well our model replicates GPT-3.5 style explanations, we used BERTScore [11], a metric designed to measure semantic similarity based on contextual embeddings. Unlike surface level comparisons like BLEU [33] or ROUGE [34], BERTScore evaluates whether two texts express similar meanings, even when phrased differently. This makes it particularly suitable for our task, where we care about capturing the same ideas, not necessarily the same words.

We applied BERTScore on a sample of 100 explanation pairs, comparing our model's outputs with the original GPT-3.5 generations for the same prompts. The mean F1 score was 0.9087, with a standard deviation of 0.014, indicating that most of the explanations are closely aligned in meaning. In practice, this means the model tends to emphasize similar themes, mention the same supporting movies, and follow the intended structure.

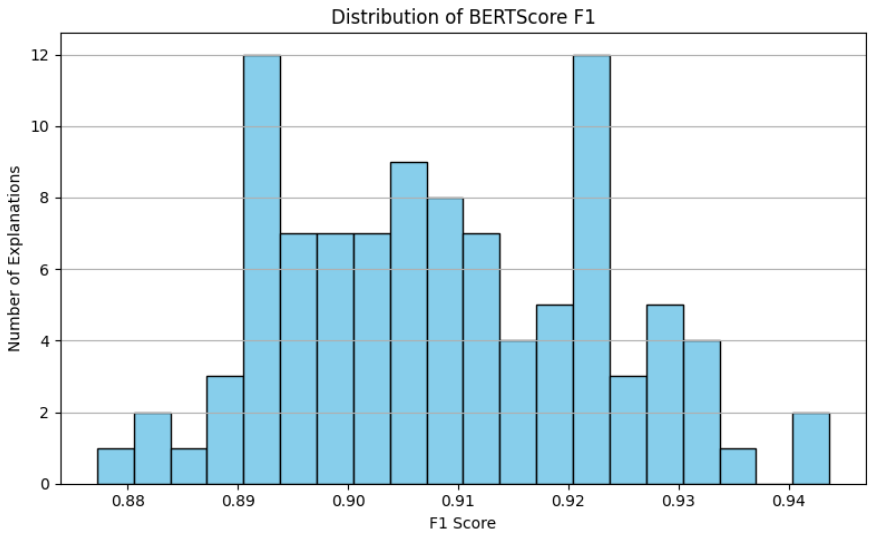

Figure 6.2: Distribution of BERTScore F1 across 100 explanations.

As shown in Figure 6.2, the F1 scores are tightly concentrated between 0.89 and 0.93, with only a few examples falling outside that range. This level of consistency suggests that the model has not only memorized examples but also learned to generalize the explanation style across cases, even with only 1,000 training samples.

Table 6.1: BERTScore summary statistics across 100 explanation pairs.

Metric	**Mean**	**Std**	**Min – Max**
Precision	0.8970	0.0143	0.8649 – 0.9311
Recall	0.9209	0.0168	0.8899 – 0.9566
F1 Score	0.9087	0.0145	0.8772 – 0.9437

As expected, the Recall scores are slightly higher than Precision, suggesting that our model tends to be a bit more concise than GPT-3.5, while still preserving the key ideas. Overall, this evaluation shows that the fine-tuned model has learned to produce explanations that are meaningfully close to the originals it was trained on.

Of course, BERTScore only captures similarity in meaning, it does not evaluate factual consistency or structure. But as a first step, it gives us confidence that the model has learned the right instincts.

6.3.2 Binary Acceptability

Although semantic similarity metrics such as BERTScore can offer useful insight into how closely a generated explanation aligns with a reference, they cannot ensure whether it actually makes sense or stays true to the user's preferences. A response may appear well-written and semantically close, yet still include false associations, irrelevant references, or fail to justify the recommendation in a meaningful way. To address these limitations, we performed a binary evaluation focused on clarity, correctness, and relevance.

For this was structured as a strict Yes/No judgment based on three straightforward criteria. Each explanation had to mention at least two movies from the user's viewing history, establish a relevant connection to the recommended title, typically through genre, plot, or director, and avoid incorrect claims or mismatched associations. We refined the prompt gradually to catch real mistakes, without letting small issues in phrasing or tone affect the outcome.

To ensure a fair judgment, the evaluator was given a structured summary of the user's viewing history and the recommended movie along with the generated explanation.

We applied this process to a random sample of 100 generated responses, using GPT-4o as the evaluator. The results are presented in Table 6.2.

Table 6.2: Results from GPT-4o binary acceptability evaluation.

Label	**Count**	**Percentage**
Acceptable (Yes)	79	79.0%
Unacceptable (No)	21	21.0%

The results indicate that most explanations were clear, relevant, and grounded in the user's viewing history. Despite the strict evaluation criteria, 79% of the outputs were rated as acceptable, suggesting that the model often generates explanations that go beyond surface similarity and reflect real connections to the user's preferences.

To illustrate how these criteria were applied in practice, we include one accepted and one rejected example, along with GPT-4o's exact responses.

Example 1 – Accepted Explanation

Model output:

> We recommend this movie because *Tough and Deadly* shares similarities with *Die Hard: With a Vengeance* and *Die Hard 2* in terms of being an action-packed thriller with a strong protagonist facing off against dangerous adversaries. Additionally, the director, Steve Cohen, brings a unique touch to the genre, blending action, drama, and thriller elements that you may enjoy based on your previous ratings.

GPT-4o evaluation:

> Yes. The explanation is acceptable as it references two movies from the user's viewing history, *Die Hard: With a Vengeance* and *Die Hard 2*, and draws a logical connection based on genre and action elements. There are no factual errors.

Example 2 – Rejected Explanation

Model output:

> We recommend this movie because *Brazil* shares a similar director, Terry Gilliam, with *Down by Law*, which you rated highly. Additionally, the sci-fi genre of *Brazil* aligns with the drama genre of *Surviving Picasso*, offering a unique blend of storytelling that you may find intriguing.

GPT-4o evaluation:

> No. The explanation incorrectly states that *Brazil* shares a director with *Down by Law*. While both films are notable, they are directed by different individuals (Terry Gilliam for *Brazil* and Jim Jarmusch for *Down by Law*). This factual error undermines the recommendation's credibility.

These examples reflect the kind of issues this evaluation is designed to uncover, from explanations that make clear, sensible connections to those that include small but important mistakes. Together with semantic similarity metrics, this layer provides a more complete view of the model's ability to produce explanations that are not only fluent, but also trustworthy.

Appendix B.1 contains the full evaluation prompt as well as representative examples of inputs, generated explanations, and GPT-4o's responses.

6.3.3 GPTScore-Style

The binary evaluation allowed us to confirm whether an explanation was logically and factually acceptable. However, not all valid outputs are equally strong. Some barely meet the basic criteria, while others offer more specific and user-tailored reasoning. To better understand this variation in quality, we carried out a second round of evaluation inspired by the GPTScore framework [12].

The goal was to examine how specifically it referred to the user's history and how clearly it communicated the reason behind the recommendation. GPT-4o received both the viewing history and the generated explanation and returned two scores from 1 to 5, one for personalization and one for helpfulness. The prompt used for this evaluation and examples is included in Appendix B.2.

Table 6.3: Results from GPT-based evaluation of personalization and clarity.

Criterion	Average Score
Specificity & Personalization	3.77 / 5
Helpfulness & Clarity	4.09 / 5

As shown in Table 6.3, the model performs well overall. Helpfulness scored 4.09 on average, indicating that most explanations were clearly phrased and easy to follow. Personalization scored slightly lower, with an average of 3.77, showing that while the model often referred to the user's history, the level of specificity varied.

To give a fuller view, Figure 6.3 shows the distribution of scores across the 100 explanations.

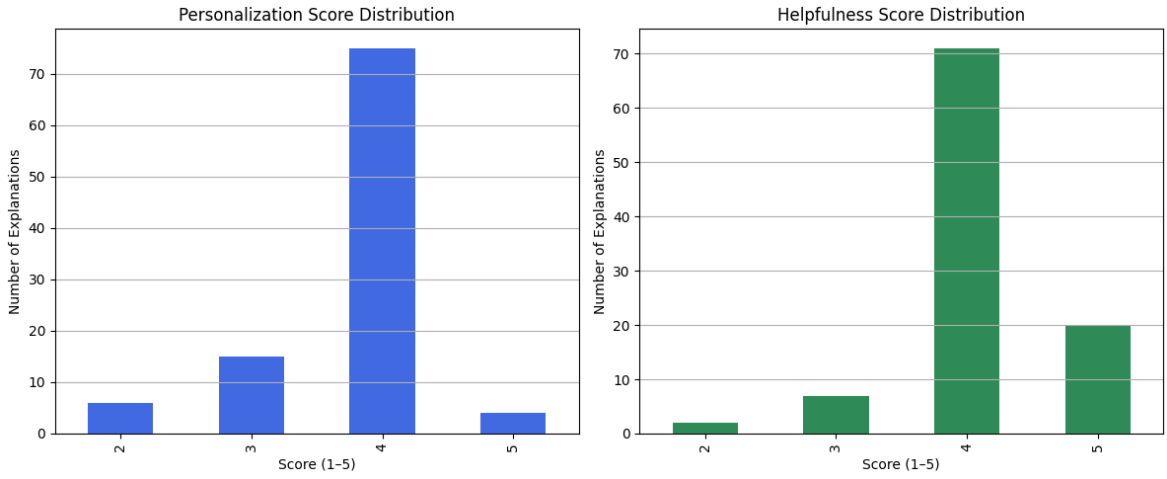

Figure 6.3: Distribution of personalization and clarity scores across 100 explanations.

Most explanations received high scores in both categories. Even in cases with lower personalization, the model still referred to movies the user had seen. However, as noted in GPT-4o's own evaluations, these references were often superficial, pointing to genres or mentioning titles without explaining how the recommended movie connects to the user's preferences. In one example, the evaluator commented that the explanation "mentions the genre but does not provide specific details about how the themes or stories relate to the user's previously watched films."

6.4 Summary and Outlook

In this chapter, we examined how language models can be used to justify recommendations in a way that feels natural and understandable. By fine-tuning a lightweight version of LLaMA 3 on a filtered dataset, we enabled the system to generate short, targeted explanations tailored to each recommendation.

The evaluation showed that, in most cases, the model produced clear and accurate justifications. It frequently drew sensible connections between movies and captured the user's preferences in a way that made sense. Only a handful of cases were more generic. Considering the small training set and the setup, the overall performance is very encouraging.

In the next chapter, we turn to conversational recommendation, extending the idea of explainability into a more interactive setting. We explore how users can engage with the system in real time, asking for recommendations, receiving justifications, and refining their preferences through dialogue.

Chapter 7

Conversational Recommender System

In this part of the work, we move beyond static recommendations and explanations to a system that handles conversation in a more open and flexible way. The aim is not simply to pair users with movies, but to develop an assistant that can interpret requests expressed in any way and keep track of the conversation as preferences shift or new questions arise.

To make this possible, we combine several key components: an intent detection module powered by a large language model, a semantic retrieval mechanism based on sentence embeddings, and a response generation pipeline. Each part is designed to work together so that the assistant can respond appropriately as the conversation progresses.

In the following sections, we break down the architecture and workflow of the conversational assistant and show how these elements interact to support a more natural approach to movie discovery.

7.1 System Architecture

We organize the conversational assistant as a modular pipeline, where each component is responsible for a different aspect of interaction. When a user sends a message, the system chooses the appropriate processing path based on the nature of the request, whether that involves retrieving relevant movies, looking up factual metadata, or generating a casual reply.

Regardless of the path taken, the final response is generated by LLaMA 3, which combines the current input with recent dialogue history to produce context-aware and coherent outputs.

Figure 7.1 summarizes how the components interact during a typical conversation.

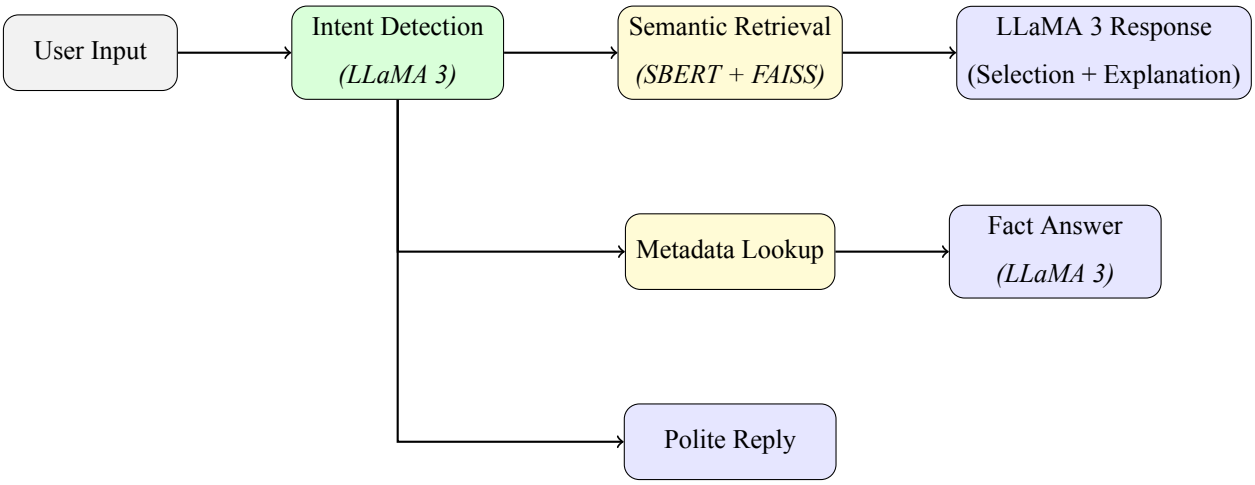

Figure 7.1: Architecture of the conversational movie recommendation assistant.

7.2 Intent Detection

Before generating a response, the system must determine whether the user is asking for a recommendation, requesting information about a movie, refining a previous suggestion, or simply engaging in casual conversation. Correct intent recognition ensures that the assistant chooses the appropriate processing path.

In our implementation, we use the LLaMA 3 language model to interpret user messages through structured prompt. Instead of fine-tuning it for classification, we treat the task as a controlled generation problem: the model receives a brief instruction describing the possible actions, together with the user's request, and is asked to return a single JSON object representing the user's intent.

We distinguish between four intent types: `"recommendation"` when users request new suggestions, `"refine_query"` if they reject or adjust a previous result, `"lookup"` for factual questions about a movie and `"chat"` for conversational responses that do not require further action.

The prompt used for intent classification is shown below. It defines the expected response format, gives clear rules for how each type of message should be handled, and, when needed, includes context from earlier in the conversation.

You are an intent classifier for a movie assistant. Follow these rules for every message:
- If the user requests a movie, a genre, or a type of movie, return:
{”action”: ”recommendation”, ”query”: ”<the user’s request or genre>”}
- If the user rejects or modifies a previous suggestion, return:
{”action”: ”refine_query”, ”query”: ”<the user’s updated request or preference>”}
- If the user asks for information about a movie, return:
{”action”: ”lookup”, ”title”: ”<movie title>”, ”info_type”: ”<director/overview/genres/etc>”}
- If the user’s message is a greeting, polite response, or short positive reply, return:
{”action”: ”chat”, ”response”: ”<an appropriate polite or friendly reply>”}
- If the user’s message is ambiguous, unclear, or does not make a specific request, return:
{”action”: ”chat”, ”response”: <friendly reply>}
Always return ONLY the JSON object, with no extra text or explanation.
Optional context:
The user previously asked for: ”{last_query}”
The assistant previously suggested: ”{last_title}”
User: {user_input}

Table 7.1 shows representative examples illustrating how the model handles different types of input.

Table 7.1: Intent detection and structured assistant response across a multi-turn dialogue.

User Message	**Detected Intent**	**Structured Action (JSON)**
Hmm, I’m not sure what I want today.	Chat	`{”action”: ”chat”, ”response”: ”...” }`
I’m in the mood for something suspenseful, maybe a good thriller.	Recommendation	`{”action”: ”recommendation”, ”query”: ”thriller”}`
Maybe something more psychological?	Refine Query	`{”action”: ”refine_query”, ”query”: ”psychological thriller”}`
Who directed American Psycho?	Lookup	`{”action”: ”lookup”, ”title”: ”American Psycho”, ”info_type”: ”director”}`
Thanks, that actually sounds great!	Chat	`{”action”: ”chat”, ”response”: ”...” }`

7.3 Retrieval and Selection

Once the user's intent has been identified as a recommendation request, either directly or through refinement, we proceed to retrieve suitable movie candidates. This step is approached semantically, using SBERT embeddings and FAISS for efficient similarity search. Each movie in our dataset is represented as a dense vector based on its textual description, as established in Section 3.3.

To find relevant options, we encode the user's query (e.g., "psychological thriller") using the same SBERT model and compare it against the full index of movie vectors using cosine similarity. The top-ranked results are returned as potential candidates. To encourage diversity and avoid repetition, we exclude previously recommended movies from the results.

The final selection among the retrieved movies is made using a second LLM call. We provide the model with a structured prompt that includes the metadata of the candidate movies and explicit instructions to select a single title and to justify its choice based only on the given information.

The prompt used for final selection is shown below:

```
You are a helpful movie assistant. The user is looking for: "<user_query>"
Here are 3 candidate movies:
A. Title: <movie_title_A> Genres: <genres_A> Director: <director_A> Plot: <plot_A>
B. Title: <movie_title_B> Genres: <genres_B> Director: <director_B> Plot: <plot_B>
C. Title: <movie_title_C> Genres: <genres_C> Director: <director_C> Plot: <plot_C>
Choose ONLY ONE movie that best fits the user's request. Respond clearly and in a friendly tone.
IMPORTANT:
- Do NOT mention the other movies.
- Do NOT invent genres, facts, or attributes not present above.
- Base your explanation strictly on the provided information.
```

7.4 Dialogue Management

To support multi-turn interaction, we maintain a lightweight memory of key elements from the conversation, including the user's most recent input, the last detected intent, the previous recommendation query, and the last suggested movie.

After each user turn, we update this internal memory with the latest information. For example, when the user requests a movie and receives a recommendation, both the original

query and the selected title are stored. If the user later replies with a non-specific message such as “something else” or “not that one,” the system uses this context to understand it as a refinement of the previous request. Similarly, if the user asks a question like “Who directed it?”, the assistant resolves the reference using the last recommended movie.

We provide this information directly into the intent detection prompt. As shown in Section 7.2, the user’s previous query and the last suggested title are included as optional fields.

Not all user messages correspond to a clear intent. The assistant must also handle informal inputs, such as greetings, acknowledgments, or uncertain replies, that do not require any action. In these cases, it defaults to polite responses or clarification prompts, preserving the flow of the conversation without triggering retrieval or recommendation logic.

7.5 System Behavior and Examples

This section highlights how the assistant behaves in realistic conversational scenarios, focusing on its ability to classify intent, follow context, and respond appropriately. Evaluating conversational systems remains a known challenge, as standard automatic metrics often fail to capture core aspects of dialogue quality, such as coherence, relevance, or user satisfaction [35]. In line with prior work, we adopt a qualitative approach and present representative examples that illustrate how the assistant behaves in context. The cases that follow cover different types of interaction, from recommendation requests and refinements to factual questions and casual replies. To facilitate observation and testing, the system was deployed in a lightweight Gradio interface [36] that allowed real-time multi-turn interaction.

7.5.1 Intent Classification in Dialogue

Effective intent recognition allows the assistant to respond in a way that aligns with the user’s goal. As described in Section 7.2, each user message is mapped to one of four structured intent types. This process is illustrated in Table 7.1, which presents multi-turn examples.

As the dialogue shows, the system correctly classifies ambiguous inputs such as “Hmm, I’m not sure what I want today” as `chat`, recognizes genre preferences like “a good thriller” as `recommendation`, and identifies follow-up refinements or information requests accordingly. Even informal replies like “Thanks” are handled gracefully, without interrupting the flow of the interaction. These examples confirm that intent classification works reliably and

contributes to logical interaction.

7.5.2 Context Awareness and Reference Resolution

Understanding what has been said in previous turns is just as important as detecting intent. The following exchange shows how the model handles references to earlier content without interrupting its course.

User: I want something mysterious and psychological.

Assistant: Hello there! I think I've found the perfect match for you! The movie that best fits your request for a "mysterious psychological" film is *The Machine* (1994)!

This French psychological thriller explores the dark and unsettling theme of an experiment gone wrong, where a psychiatrist's obsession with the link between mind and spirit leads to a terrifying outcome. The plot is full of suspense and intrigue, making it a thrilling and thought-provoking watch. I think you'll find it's a great fit for your request.

User: Who directed it?

Assistant: The director of the film *The Machine* (1994) is François Dupeyron.

The model understands that "it" refers to the previously mentioned film and provides the right answer without needing repetition. This contributes to a more natural interaction.

7.5.3 Refinement Handling

Users often adjust their preferences as the conversation unfolds. The next example highlights how a revised request is interpreted as a refinement.

User: Can you recommend a serious, emotionally-driven drama?

Assistant: I'd be happy to help you find a movie that fits your request!

Based on your search for a serious, emotionally-driven drama, I think the movie that best fits your request is: *Lilian's Story* (1995). This movie has a strong focus on emotional depth and drama, with a unique and powerful

storyline about a woman who's been locked up for 40 years and her journey to find love and happiness. The plot explores themes of self-discovery, relationships, and personal growth, which suggests a high level of emotional intensity. The film's tone aligns well with your request.

User: Maybe something with a historical setting?

Assistant: I've got just the one for you!
I highly recommend *The English Patient* (1996). With its historical setting in the 1930s, it's a serious and emotionally-driven drama that explores themes of love, betrayal, and politics during a tumultuous time. The film's epic narrative is set against the backdrop of World War II, adding depth and complexity to the story. Anthony Minghella's direction masterfully weaves together the characters and emotions, making it a powerful and unforgettable cinematic experience.

Rather than treating the second message as a new query, the reply incorporates the additional constraint and adjusts the recommendation accordingly.

7.5.4 Handling Informal or Ambiguous Messages

Not every message requires action. Sometimes the user is unsure, non-committal, or just polite. Even in these moments, the system responds with the right tone and without pushing the conversation forward.

User: Hmm, not sure what I feel like today.

Assistant: Feel free to give me a genre, a director, or a specific movie title, and I can help you find something that suits your mood!

User: Thanks, maybe later.

Assistant: You're welcome! Feel free to come back when you're ready for a movie recommendation.

7.6 Discussion

In this chapter, we explored how a conversational assistant can support interactive movie recommendations by combining intent detection, semantic retrieval, and natural language generation. The system adapts to different kinds of input, whether users express a clear request, refine a previous one, or simply respond in a conversational tone. By keeping track of recent dialogue elements, it preserves context and produces consistent replies across turns. While the architecture is relatively simple, the examples we presented suggest that it can support natural and responsive interactions without requiring complex dialogue management.

Chapter 8

Conclusions

The goal of this thesis was to strengthen recommendation systems by integrating large language models in a way that improves accuracy, enables explainability, and supports natural user interaction. For each of these challenges, we proposed lightweight and effective solutions, which were implemented, thoroughly analyzed, and evaluated across multiple dimensions.

Beginning with predictive accuracy, we introduced a hybrid recommendation model that combines collaborative filtering with a proposed SBERT-based model. The hybrid model consistently outperforms each individual model and maintains high performance even in sparse user–item conditions.

Turning to explainability, we created a structured dataset of user-specific prompts and used it to fine-tune an LLaMA 3 model with LoRA adapters. The result is a system that can generate fluent, personalized justifications that reflect actual user preferences.

Finally, focusing on interactive recommendation, we proposed an LLM-powered conversational assistant that engages with users through natural dialogue to provide suggestions, refinements, and explanations. Despite its simplicity, the system delivers high-quality responses, as demonstrated in a series of representative dialogue examples.

Overall, the proposed methods are modular, efficient, and capable of running on local hardware, confirming that advanced, user-focused recommendation systems can be implemented in practice.

Future work could explore alternative ensemble strategies or enrich the hybrid model with features such as temporal dynamics, user demographics, or contextual signals. While SBERT has proven highly effective in capturing semantic similarity, it may also be useful to

experiment with other embedding models, such as GTR [37] or E5 [38], to explore potential complementarities. For both explainability and interaction, future directions may include learning from user feedback and modeling long-term user behavior, aiming to support more adaptive and engaging recommendation experiences.

Bibliography

[1] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10*, pages 285–295. ACM, 2001.

[2] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[3] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM*, pages 1441–1450. ACM, 2019.

[4] Gopal Behera, Sanjaya Kumar Panda, Meng-Yen Hsieh, and Kuan-Ching Li. Hybrid collaborative filtering using matrix factorization and xgboost for movie recommendation. *Comput. Stand. Interfaces*, 90:103847, 2024.

[5] Kulvinder Singh, Sanjeev Dhawan, and Nisha Bali. An ensemble learning hybrid recommendation system using content-based, collaborative filtering, supervised learning and boosting algorithms. *Autom. Control. Comput. Sci.*, 58(5):491–505, 2024.

[6] Robin D. Burke. Hybrid recommender systems: Survey and experiments. *User Model. User Adapt. Interact.*, 12(4):331–370, 2002.

[7] Qiyao Ma, Xubin Ren, and Chao Huang. Xrec: Large language models for explainable recommendation. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 391–402. ACL, 2024.

[8] Mengyuan Yang, Mengying Zhu, Yan Wang, Linxun Chen, Yilei Zhao, Xiuyuan Wang, Bing Han, Xiaolin Zheng, and Jianwei Yin. Fine-tuning large language model based explainable recommendation with explainable quality reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9250–9259. AAAI Press, 2024.

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Xiaoyu Zhang, Yishan Li, Jiayin Wang, Bowen Sun, Weizhi Ma, Peijie Sun, and Min Zhang. Large language models as evaluators for recommendation explanations. In *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys*, pages 33–42. ACM, 2024.

[11] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020.

[12] Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. Gptscore: Evaluate as you desire. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL*, pages 6556–6576. ACL, 2024.

[13] Kun Zhou, Xiaolei Wang, Yuanhang Zhou, Chenzhan Shang, Yuan Cheng, Wayne Xin Zhao, Yaliang Li, and Ji-Rong Wen. Crslab: An open-source toolkit for building conversational recommender systems. *arXiv preprint arXiv:2101.00939*, 2021.

[14] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining*, pages 304–312. ACM, 2020.

[15] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In

The Eleventh International Conference on Learning Representations, ICLR. OpenReview.net, 2023.

[16] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian J. McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems. In *Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR, Proceedings, Part II*, volume 14609 of *Lecture Notes in Computer Science*, pages 364–381. Springer, 2024.

[17] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*, pages 3980–3990. ACL, 2019.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4171–4186. ACL, 2019.

[19] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

[20] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR*, pages 39–48. ACM, 2020.

[21] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 4512–4525. ACL, 2020.

[22] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020.

[23] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. Springer, 2011.

[24] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[25] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW*, pages 173–182. ACM, 2017.

[26] Miha Grcar, Dunja Mladenic, Blaz Fortuna, and Marko Grobelnik. Data sparsity issues in the collaborative filtering framework. In *Advances in Web Mining and Web Usage Analysis, 7th International Workshop on Knowledge Discovery on the Web, WebKDD*, volume 4198 of *Lecture Notes in Computer Science*, pages 58–76. Springer, 2005.

[27] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 4765–4774, 2017.

[28] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2023.

[29] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022.

[30] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP*, pages 38–45. ACL, 2020.

[31] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

[32] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

[33] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. ACL, 2002.

[34] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. ACL, 2004.

[35] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2122–2132. ACL, 2016.

[36] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019.

[37] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 9844–9855. ACL, 2022.

[38] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2024.

APPENDICES

Appendix A

Prompt for Generating Explanations

The following is a complete example of a prompt used to generate a natural language explanation. It includes metadata for three previously rated movies and one recommended movie, followed by a structured instruction.

You are a friendly and knowledgeable movie recommendation assistant.
Here is what the user has previously watched and rated:
1. Home Alone (1990) - Genres: Children's|Comedy - Director: Chris Columbus - Rating: 2 - Similarity to recommended: 0.4817 - Summary: Eight-year-old Kevin McCallister makes the most of the situation after his family unwittingly leaves him behind when they go on Christmas vacation. When thieves try to break into his home, he puts up a fight like no other.
2. Welcome to the Dollhouse (1995) - Genres: Comedy|Drama - Director: Todd Solondz - Rating: 5 - Similarity to recommended: 0.4815 - Summary: An unattractive 7th grader struggles to cope with suburban life as the middle child with inattentive parents and bullies at school.
3. Mallrats (1995) - Genres: Comedy - Director: Kevin Smith - Rating: 4 - Similarity to recommended: 0.4801 - Summary: Both dumped by their girlfriends, two best friends seek refuge in the local mall. Eventually, they decide to try and win back their significant others and take care of their respective nemeses.
Now you are recommending the following movie:
North (1994) - Genres: Comedy - Director: Rob Reiner - Summary: Eleven-year-old North has had it with his parents. They are always busy with their careers and don't give North the attention he needs, so he files a lawsuit against them. The judge rules that North should either find new parents or return to his own parents within two months. Thus North starts off on a journey around the world to find parents that really care about him.

Write exactly two sentences that explain to the user why this movie is recommended.
Start with: *"We recommend this movie because..."*

Be specific and natural—refer clearly to at least two previously watched movies, focusing on similarities in genre, story, or director.
If the user rated similar movies highly (≥4), emphasize the match. If they rated them poorly (≤2), politely explain why this one might still appeal.

Avoid vague phrases, be clear, personal, and sound like a helpful friend. Output only the two-sentence explanation—nothing else.

Appendix B

Explanation Evaluation: Prompts and Examples

This appendix contains the full prompts used for the automated evaluation of explanation quality, along with representative examples. The same explanations are reused across both evaluation methods.

B.1 Binary Acceptability

The following prompt was used in the binary acceptability evaluation. It instructed GPT-4o to make a simple Yes/No judgment based on logical and factual correctness.

> You are reviewing a movie recommendation explanation written by an assistant. Your goal is to check whether the explanation is factually and logically sound — not whether it's perfectly written or highly detailed. The explanation should be marked as ACCEPTABLE if:
> - It mentions at least two movie from the user's viewing history.
> - It draws a connection in genre, theme, or narrative logic — even if the match isn't perfect.
> - It does not make obvious factual errors (e.g., wrong genre, invented plots).
>
> Use common sense and be forgiving of small imperfections. If the explanation makes an honest and reasonable attempt to justify the recommendation based on the user's past preferences, it's acceptable. Only if there is a clear factual mistake, made-up detail, or reasoning that doesn't follow at all, answer: No. Otherwise, answer: Yes. After your Yes/No, briefly explain your decision, focusing only on factual or logical correctness.
>
> User history and recommendation: {prompt_text}
>
> Assistant's explanation: {explanation}. Is the explanation factually and logically correct?

Example 1 – Accepted Explanation

Prompt_text:	1. *Die Hard: With a Vengeance* (1995) - Genres: Action|Thriller - Director: John McTiernan - Rating: 5 - Similarity to recommended: 0.5939 - Summary: New York detective John McClane is back and kicking bad-guy butt in the third installment of this action-packed series, which finds him teaming with civilian Zeus Carver to prevent the loss of innocent lives. McClane thought he'd seen it all, until a genius named Simon engages McClane, his new "partner" – and his beloved city – in a deadly game that demands their concentration. 2. *Hackers* (1995) - Genres: Action|Crime|Thriller - Director: Iain Softley - Rating: 3 - Similarity to recommended: 0.5915 - Summary: Along with his new friends, a teenager who was arrested by the US Secret Service and banned from using a computer for writing a computer virus discovers a plot by a nefarious hacker, but they must use their computer skills to find the evidence while being pursued by the Secret Service and the evil computer genius behind the virus. 3. *Die Hard 2* (1990) - Genres: Action|Thriller - Director: Renny Harlin - Rating: 4 - Similarity to recommended: 0.5824 - Summary: One year after his heroics in Los Angeles, John McClane is an off-duty cop who is the wrong guy in the wrong place at the wrong time. On a snowy Christmas Eve, as he waits for his wife's plane to land at Washington Dulles International Airport, terrorists take over the air traffic control system in a plot to free a South American army general and drug smuggler being flown into the US to face drug charges. It's now up to McClane to take on the terrorists, while coping with an inept airport police chief, an uncooperative anti-terrorist squad, and the life of his wife and everyone else trapped in planes circling overhead. **Recommended movie:** *Tough and Deadly* (1995) - Genres: Action|Drama|Thriller - Director: Steve Cohen - Summary: A private eye helps an amnesiac CIA agent elude mobsters who don't want him to regain his memory of their drug operation.
Explanation:	We recommend this movie because Tough and Deadly shares similarities with Die Hard: With a Vengeance and Die Hard 2 in terms of being an action-packed thriller with a strong protagonist facing off against dangerous adversaries. Additionally, the director Steve Cohen brings a unique touch to the genre, blending action, drama, and thriller elements that you may enjoy based on your previous ratings.
GPT-4o Evaluation:	Yes. The explanation is acceptable as it references two movies from the user's viewing history, *Die Hard: With a Vengeance* and *Die Hard 2*, and draws a logical connection based on genre and action elements. There are no factual errors.

Example 2 – Rejected Explanation

Prompt_text:	1. *Down by Law* (1986) - Genres: Comedy|Drama - Director: Jim Jarmusch - Rating: 4 - Similarity to recommended: 0.4618 - Summary: A disc jockey, a pimp and an Italian tourist escape from jail in New Orleans. 2. *Blow-Out (La Grande Bouffe)* (1973) - Genres: Drama - Director: Marco Ferreri - Rating: 5 - Similarity to recommended: 0.4423 - Summary: Four friends gather at a villa with the intention of eating themselves to death. 3. *Surviving Picasso* (1996) - Genres: Drama - Director: James Ivory - Rating: 3 - Similarity to recommended: 0.4388 - Summary: The passionate Merchant-Ivory drama tells the story of Francoise Gilot, the only lover of Pablo Picasso who was strong enough to withstand his ferocious cruelty and move on with her life. **Recommended movie:** *Brazil* (1985) - Genres: Sci-Fi - Director: Terry Gilliam - Summary: Low-level bureaucrat Sam Lowry escapes the monotony of his day-to-day life through a recurring daydream of himself as a virtuous hero saving a beautiful damsel. Investigating a case that led to the wrongful arrest and eventual death of an innocent man instead of wanted terrorist Harry Tuttle, he meets the woman from his daydream, and in trying to help her gets caught in a web of mistaken identities, mindless bureaucracy and lies.
Explanation:	We recommend this movie because Brazil shares a similar director, Terry Gilliam, with Down by Law, which you rated highly. Additionally, the sci-fi genre of ”Brazil” aligns with the drama genre of ”Surviving Picasso,” offering a unique blend of storytelling that you may find intriguing.
GPT-4o Evaluation:	No. The explanation incorrectly states that *Brazil* shares a director with *Down by Law*. While both films are notable, they are directed by different individuals (Terry Gilliam for *Brazil* and Jim Jarmusch for *Down by Law*). This factual error undermines the recommendation’s credibility.

B.2 GPTScore-Style

In this second setup, GPT-4o was instructed to rate each explanation along two dimensions, *Specificity & Personalization* and *Helpfulness & Clarity*, with short justifications. The full prompt is shown below.

You are an expert movie evaluator. Your task is to judge the quality of the following explanation, given the user's viewing history.
You must rate each of the following:
1. Specificity & Personalization: Does it refer to specific movies, ratings, or details from the user's history? (Rate 1–5 + reasoning)
2. Helpfulness & Clarity: How helpful or clear is the explanation for the user? (Rate 1–5 + reasoning)
Return your answer in this format:
{{"specificity_personalization": 1–5,
"specificity_reason": "...",
"helpfulness_clarity": 1–5,
"helpfulness_reason": "..."}}
User history: {prompt_text}
Assistant's explanation: {explanation}

The evaluations below correspond to the same examples shown earlier.

Example 1	*specificity_personalization:* 4
GPT-4o Scores	*specificity_reason:* "The explanation specifically references 'Die Hard: With a Vengeance' and 'Die Hard 2', which are both movies the user has rated highly, and it connects the recommendation to the user's viewing history effectively."
	helpfulness_clarity: 5
	helpfulness_reason: "The explanation is clear and concise, directly addressing the user's interests in action-packed thrillers and providing a logical reason for the recommendation based on their past ratings."
Example 2	*specificity_personalization:* 2
GPT-4o Scores	*specificity_reason:* "The explanation mentions 'Down by Law' and 'Surviving Picasso,' but it incorrectly states that both are directed by the same person, which is not true. It also does not accurately reflect the user's viewing history in terms of genre alignment, as 'Brazil' is a sci-fi film and the user has not rated any sci-fi films."
	helpfulness_clarity: 3
	helpfulness_reason: "While the explanation attempts to connect the recommended movie to the user's history, the inaccuracies may confuse the user. The recommendation lacks clarity due to the incorrect information about the director and the genre alignment."