# ReelReason:
## A LLM-Driven Conversational Movie Recommendation Platform

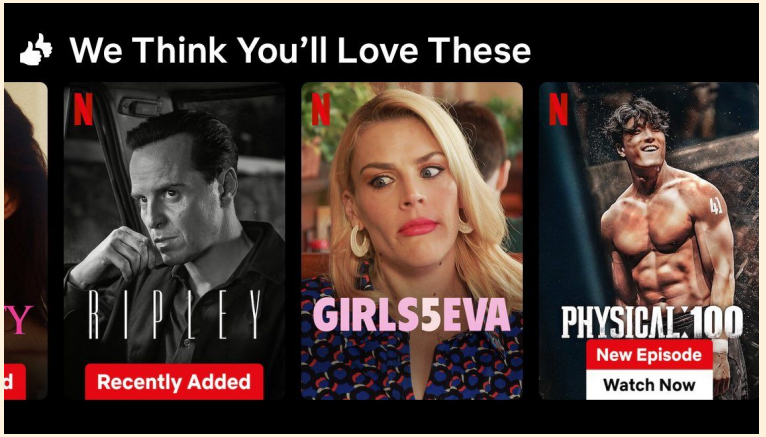

Nikhil Sharma (ns3942)
Eesun Moon (em3907)
Emily Wang (eaw2233)

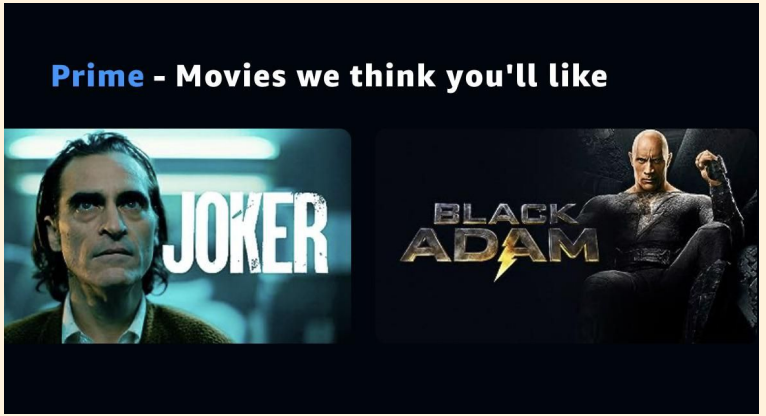
# Problem Statement & Motivation

- **The Black Box Problem:** Modern streaming platforms recommend content with limited to no explanation. Users see what to watch but never why to watch

- **Even "AI-Powered" Falls Short**: Emerging AI recommendation tools still prioritize finding the "right answer" over transparency.

- **Why Now:** Conversational explainability is very popular within modern AI, and entertainment recommendations affect millions daily, from casual viewers to seasoned cinephiles



# Research Questions & Scope

**RQ 1** How can we encode user "taste" from heterogeneous data: ratings, metadata, and natural dialogue?

**RQ 2** Can LLM reasoning improve both recommendation retrieval quality and generate meaningful explanations?

**RQ 3** Can we present info in a way that allows users to interactively find and redefine their taste profile through feedback and conversation?

## In-Scope:

- Individual-centered personalization of recs
- Taste Embeddings generation and visualization
- Conversational explainability of decisions made by system

## Out-of-Scope:

- User-to-user matching
- Supporting LetterBoxd or IMDB profiles (for now)
- Collaborative filtering (similar-user based recommendations)

# Dataset

## Movie-level

**MovieLens 1M**
(Ratings + metadata)

**MovieTweetings**
(Twitter-based ratings)

**INSPIRED Movie DB**
(dialogue-relevant metadata)

**+TMDB API: Enrichment**
(overview, genres, keywords, cast, runtime)

## Dialogue-level

**ReDial**
(Conversational movie recommendation)

**CCPE**
(dialog interactions with preference cues)

**INSPIRED Dialogues**
(speaker-dependent conversations)

# Preprocessing Pipeline

## Movie-level
(MovieLens, Tweetings, INSPIRED)

**Normalization**
: Normalize column formats across data

**Cleaning**
: remove duplicates, fix missing Movie IDs

**TMDB-enrich**
: add metadata based on Movie IDs/name

## Dialogue/User-level
(ReDial, CCPE, Inspired-dialogue)

**Processed Text**
: Clean/standardize text fields + Extract utterances

**Normalization**
: Normalize Speaker identifiers

**Flattenation**
: long conversations → single profiles/user

# Dataset Summary

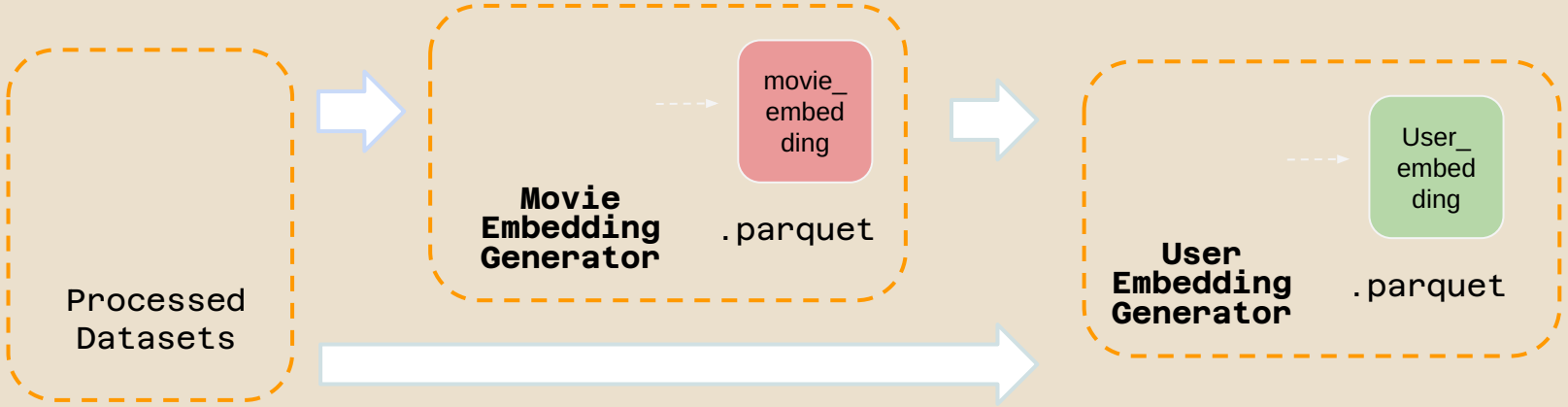
**Movie-level**
(MovieLens, Tweetings, INSPIRED)

**Dialogue/User-level**
(ReDial, CCPE, Inspired-dialogue)

```
=== Dataset Summary (Plain Text) ===
- MovieLens Movies (TMDB enriched)          [movie]  -> 3,883 rows  (movielens_movies_tmdb.csv)
- MovieTweetings Movies (TMDB enriched)     [movie]  -> 38,018 rows  (movietweetings_movies_tmdb.csv)
- INSPIRED Movie Database (TMDB enriched)   [movie]  -> 17,869 rows  (inspired_movie_database_tmdb.csv)
- MovieLens Ratings                         [rating]  -> 1,000,209 rows  (movielens_ratings.csv)
- MovieTweetings Ratings                    [rating]  -> 921,398 rows  (movietweetings_ratings.csv)
- ReDial Dialogues                          [dialogue]  -> 206,102 rows  (redial_dialogues.csv)
- CCPE Dialogues                            [dialogue]  -> 11,971 rows  (ccpe_dialogues.csv)
- GoEmotions Texts                          [text]  -> 211,225 rows  (goemotions_text_emotions.csv)

=== Totals by Category ===
- movie    : 59,770
- rating   : 1,921,607
- dialogue: 218,073
- text     : 211,225
```

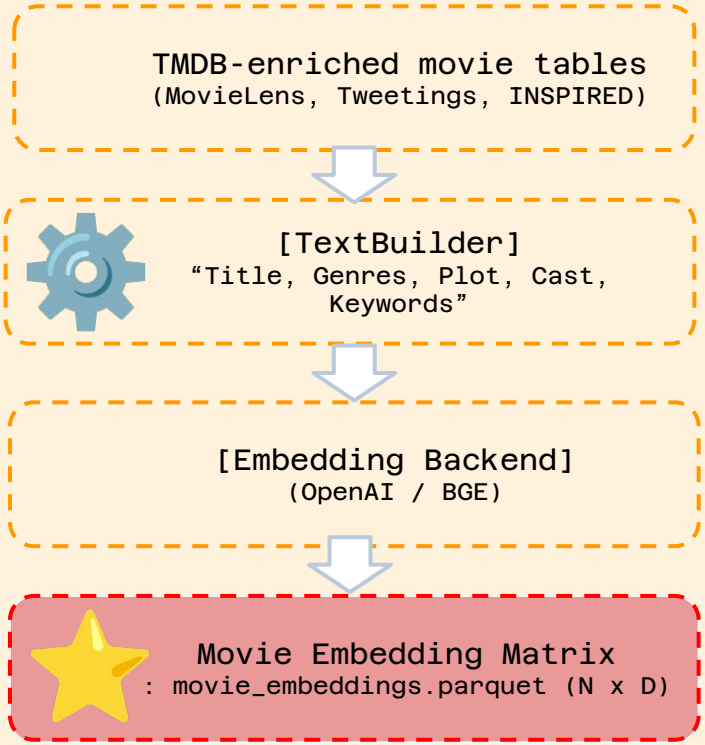
Total rows across all available datasets: **2,410,675** ✨




# Taste Embedding Generator

# Movie Embedding Generator

: Convert multi-source movie metadata into a unified semantic embedding space




**Step 1**: Build Rich Text Description per Movie
- Constructs a unified textual representation using columns
  - “Title + Year”, “Genres”, “Plot(overview)”, “Cast”, “Director (INSPIRED)”, “Keywords”
  - (Example format) “Title: inception (2010). Genres: Action, Sci-Fi. Plot: .., Cast: .., Keywords: ..”

**Step 2**: Handle Duplicates Across Datasets
- Ensures identical movies across datasets use one embedding vector

**Step 3**: Embedding with Backend (Batching)
- SentenceTransformer (v1): *BAAI/bge-base-en-v1.5*
  - Produces normalized embeddings (cosine similarity)
  - Fast local embedding
- OpenAI (v2): *text-embedding-3-large*
  - Handles long overviews with truncation
  - Batched API calls

# User Embedding Generator

: Represent user taste using ratings + dialogue signals

Rating-Based Embeddings
(MovieLens Ratings)

Text-Based Embeddings
(Dialogs: ReDial, CCPE, INSPIRED)

📍**Formula**: embedding = α * embedding_rating + **(1 - α)** * embedding_text

📍 Pipeline

**Step 1**: Select liked movies
- Rating >= 4.0 (threshold adjustable)
- **Group by user**

**Step 2**: Convert liked movies to embedding
- Filters: only users with >=3 liked movies included

**Step 3**: Map each MovieID → Movie embedding vector & Compute mean embedding per user

**α-Mix** to Create Final User Embedding

User Embedding Matrix

📍 Pipeline

**Step 1**: Collect all text utterances from user/dialog
- Identifier: user_id, dialog_id

**Step 2**: Concatenate to a long profile string

**Step 3**: Embed using same backend model

# Embedding Quantitative Analysis

: (1) **Genre Separation Gap** (Neighborhood Quality),
(2) **HitRate@10** (Recommendation Capability)

| **Metric** | **BGE-base (v1)** | | **OpenAI (v2)** | |
|---|---|---|---|---|
| | Movie | User | Movie | User |
| **Embedding Shape** (count, dim) | (59770, 768) | (9647, 768) | (59770, 3072) | (9647, 3072) |
| **Genre Separation Gap (*Drama)**: same - diff | 0.0151 | | 0.0256 | |
| **HitRate@10 : hits/total** | 0.0300 : 15/ 500 | | **0.1640** (5x) : 82 / 500 | |

**Genre separation:** Absolute similarity values are not comparable across models, so we examine the *gap*. Because genres are multi-label (e.g., “Drama|Romance”, “Comedy|Drama”), genre purity is low, leading to small gaps for both models. Still, OpenAI shows a **larger separation gap** than HuggingFace.

**HitRate@10 meaning:** A hit occurs when at least one true positive movie (rating ≥ 4) appears in a user's Top-10 recommendations.

- **OpenAI HR@10 = 0.164**, which is strong for a pure content-based system and far above random (~0.002).
- **HuggingFace HR@10 = 0.03**, indicating much weaker user–movie alignment.

# Recommender Backend

# Backend - Architecture


"My top movies are…"

"I like sad movies"

GPT call to normalize chat input (e.g. "sad movies really break my heart, I always remember those the best" → "user likes sad movies")

OpenAI text-embedding-3-large or bge-base

Embeds user input to match embeddings format

Top 20 movies by cosine similarity retrieved

Fine-tuned GPT chooses top 5 out of 20, generates explanations

📍 **Key Technologies**

- Movie embeddings data
- SentenceTransformers (BGE-base)/OpenAI
- Custom Vector Index
- OpenAI GPT (explanations + fine-tuned re-ranker)
- Persistent User State + Interaction Logs

# User Taste Representation + Multi-Turn Memory

## Taste Embedding

- Raw user input → text-embedding-3-large/bge-base embeddings
  a. Same semantic space as movie database embeddings
- Can be in any form - list of liked movies, genres,
- Optional Improvement - using lightweight GPT to convert conversation input first for cleaner semantic matching

## Preference Memory

- Exponential moving average:
  - 80% previous taste
  - 20% new input
- Taste vector evolves with the conversation

My favorite movies are Inception, Goodfellas, and WALL-e

You enjoy thrillers and sci-fi. Based on these preferences, I think you would like… here's' why…

I also have a soft spot for siblings in movies.

# Recommendation Selection

**Precision Reranking (Top 5)**

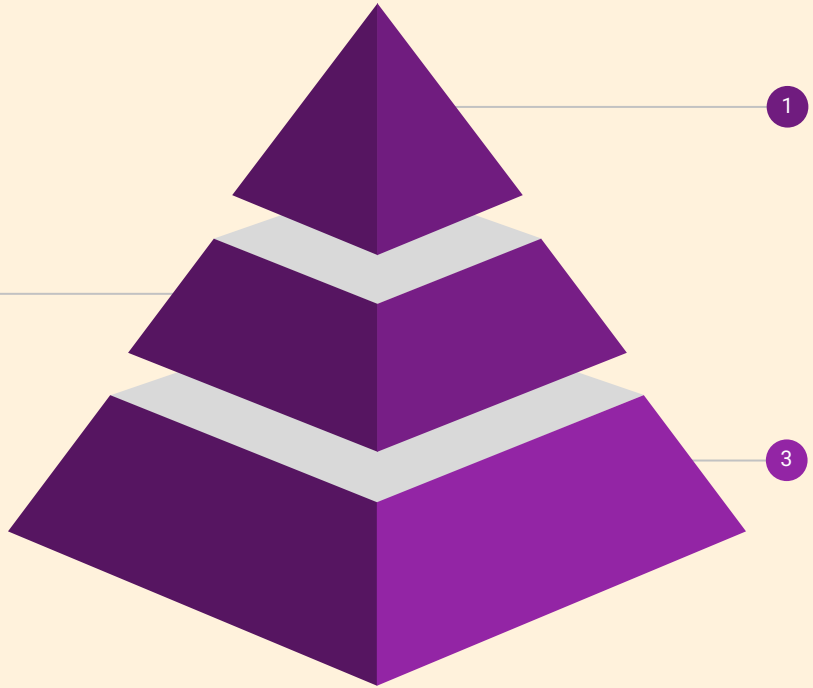
Fine-tuned GPT model conditions on taste memory and candidate metadata to select top 5 options and generate explanations

**Top 20 from Vector Index Search**

Uses Nearest-Neighbors Vector Search (faiss IndexFlatIP) across movies database

Cosine similarity metric

Match movie embeddings with user taste vector, retrieve closest 20 matches




**Full Movie Database + User preference info**

text-embedding-large/bge embed

User chats saved into exponential moving average taste vector

Optional: GPT simplification of user chats pre-embedding

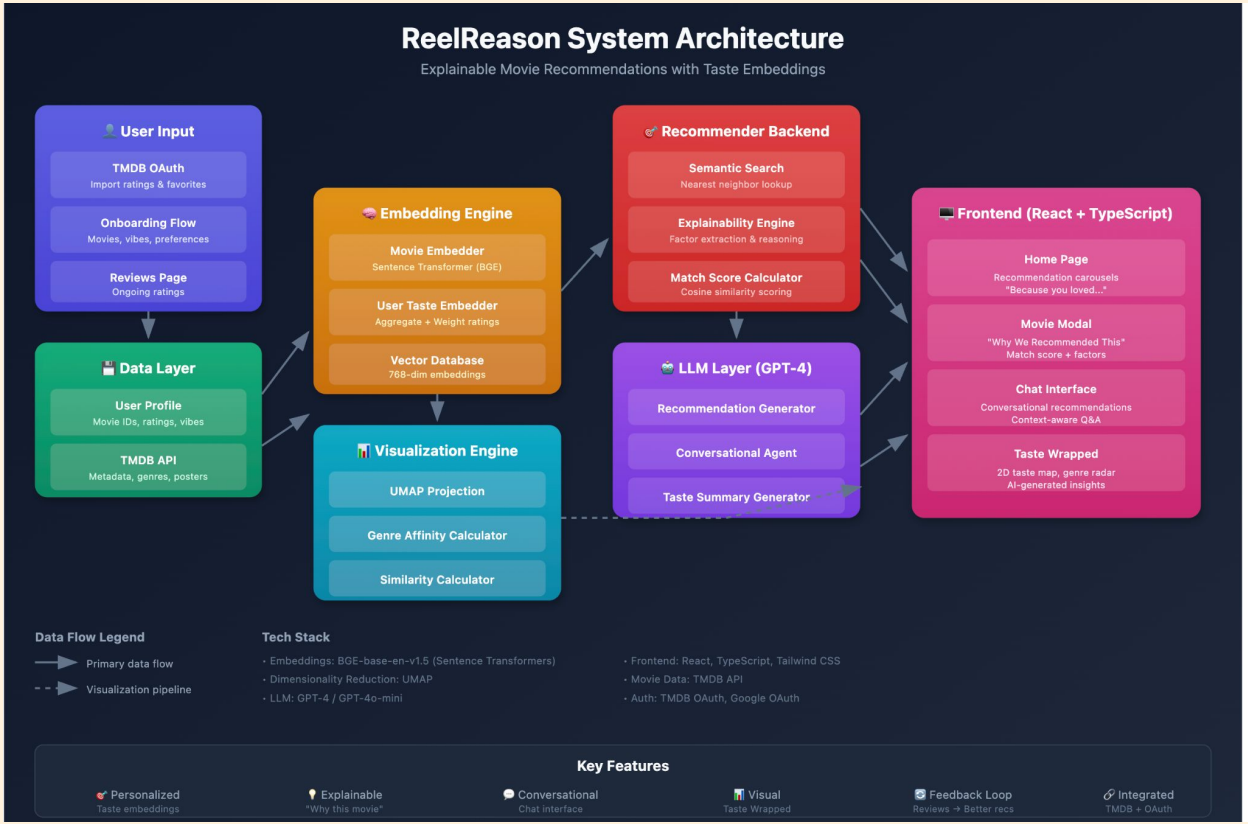
# Fine-tuning GPT for recommender quality

- **Data used**
  - Ran a capped DFS using the Letterboxd API
  - Obtained user favorites, rating history, review text
  - Used to fine-tune GPT, learn rating-based preference patterns
- **Data processing**
  - Created user taste embedding from their top 4 movies and other highly-rated movies with review text

- **Input**: user taste embedding + candidate movie
- **Output**: predicted enjoyment of the movie, from 1-5
- Ground truth is the Letterboxd rating of candidate movie
- Improves the LLM-based reranking step of selecting top 5 out of 20 retrieved movies + improved personalization on explanations

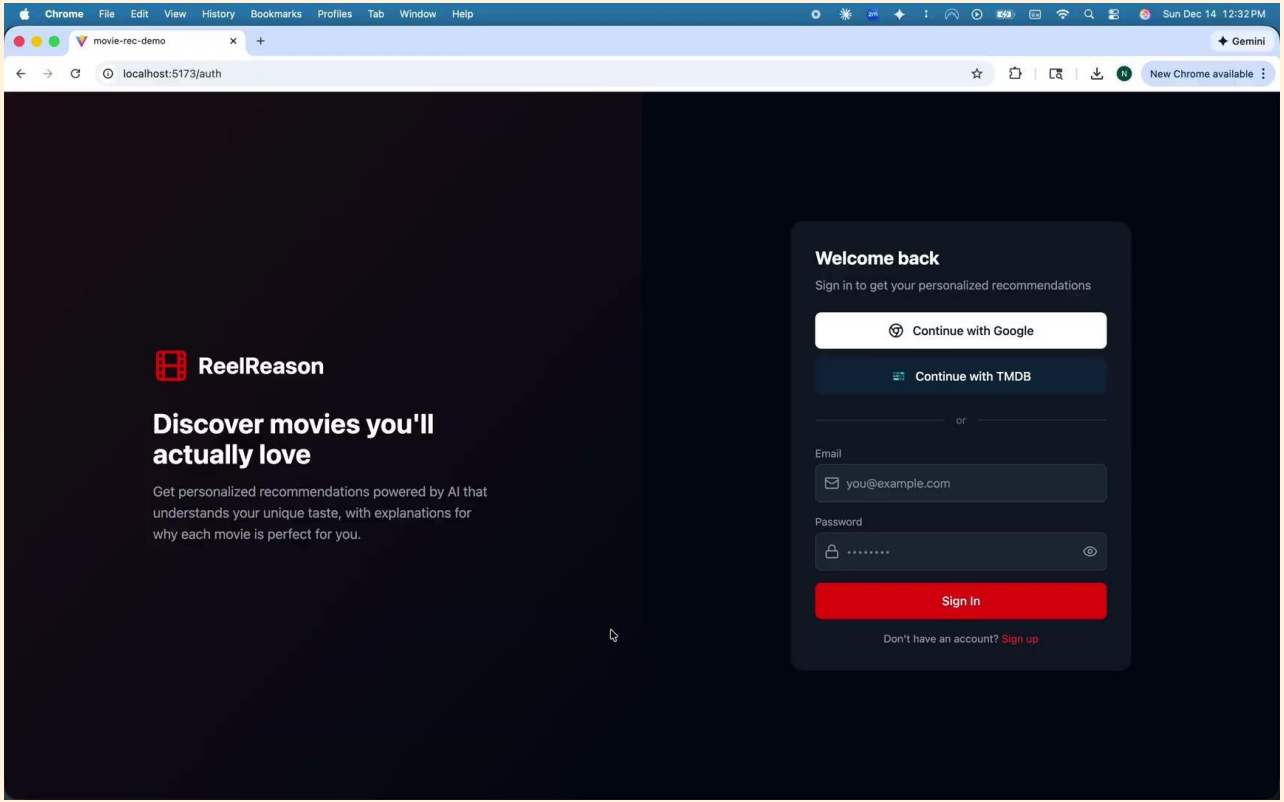
# Front-end

# Architecture Diagram



# Front-end Demo

https://drive.google.com/file/d/1jK6DhKRHUEzaLqlkGZOky77W_oEu6VMR/view?usp=sharing



# Embeddings Eval

- **Metric**: mean cosine similarity b/w user taste embedding, movie embeddings of final 5 recommendations
- 4o-based system evaluation of recommendation/interaction quality
- pre-fine tuning

| Embeddings Version | Mean Cos Sim | Std Dev | #Pairs |
|---|---|---|---|
| **BGE-base embeddings** | 0.781 | 0.084 | 1250 |
| **OpenAI embeddings** | 0.796 | 0.081 | 1250 |

| Metric | bce-base | openai | Δ |
|---|---|---|---|
| **Relevance** | 3.99 | 4.10 | +0.11 |
| **Diversity** | 3.91 | 4.05 | +0.14 |
| **Personalization** | 3.95 | 3.95 | +0.00 |
| **Explanation Quality** | 4.43 | 4.45 | +0.02 |
| **Overall Satisfaction** | 4.0 | 4.05 | +0.05 |

- **OpenAI embeddings performed better**

# Quantitative Results (fine-tuning)

- Using openai embeddings

| System Version | Mean Cos Sim | Std Dev | #Pairs |
|---|---|---|---|
| **Baseline (embeddings)** | 0.742 | 0.091 | 1250 |
| **Fine-tuned LLM** | **0.768** | 0.086 | 1250 |

- **+0.026 increase** in semantic alignment
- Modest improvement from fine-tuned reranking

# Qualitative Results: Fine-tuning Impact

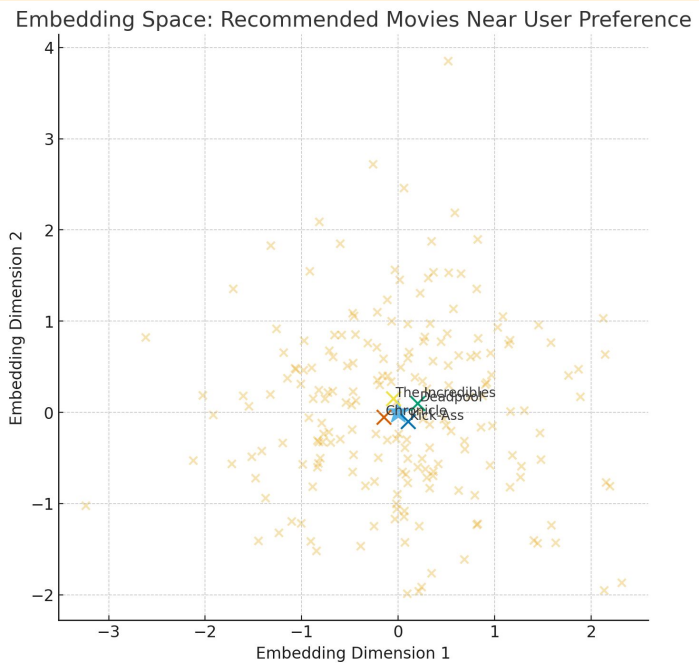
- GPT scores chat recommendation quality on letterboxd profiles
- Rated on five axes

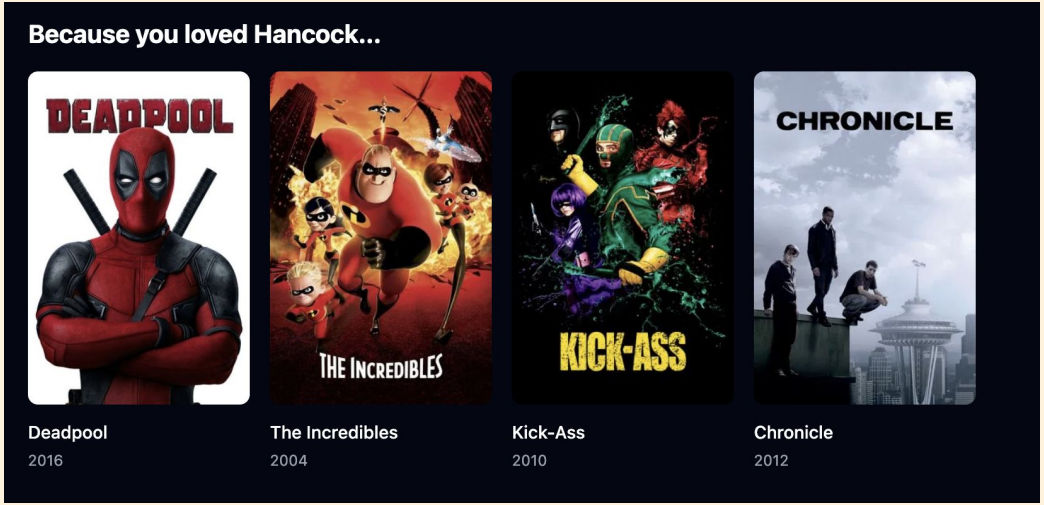
| Metric | Before FT | After FT | Δ |
|---|---|---|---|
| **Relevance** | 4.10 | **4.35** | +0.25 |
| **Diversity** | 4.05 | **4.10** | +0.05 |
| **Personalization** | 3.95 | **4.25** | +0.30 |
| **Explanation Quality** | 4.45 | **4.60** | +0.15 |
| **Overall Satisfaction** | 4.05 | **4.32** | +0.27 |

- Best increases in personalization, relevance
- Diversity stable
- **Note**: GPT-based eval can be prone to subjectivity and hallucination. With more time, we would have added more human evals to run a full validation of this LLM-based evaluator

# Qualitative Results: Error Analysis

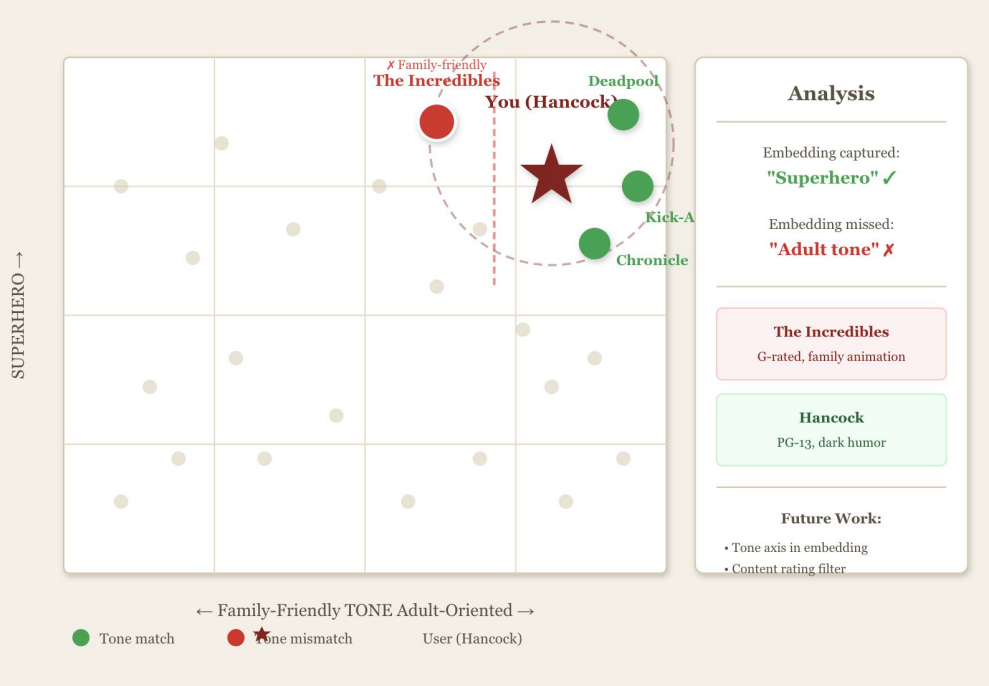
- One of these recommended movies is not like the others…
- Part of future work, better incorporate “maturity” level of movies



# Future Work

- Enhance database for more movie recommendations
- Fine-tune the embedding model for end-to-end task alignment
- Better address “tone” of recommended movies
- Expand suggestions to TV shows as well



# Ask us anything!

# Appendix

# GPT Qualitative Prompt

```
"""You are evaluating a movie recommendation assistant.

You will be given a short conversation between a USER and ASSISTANT.
The pattern is:
- USER expresses their tastes or constraints
- ASSISTANT provides a set of movie recommendations
- USER refines their preferences
- ASSISTANT updates the recommendations.

Your task:
1. Read the entire conversation.
2. Rate the assistant's performance on:
  - relevance: how well do the recommendations match the user's tastes?
  - diversity: how diverse are the recommendations while still matching preferences?
  - personalization: how well does the assistant adapt to new user feedback across turns?
  - explanation_quality: how clear and convincing are the explanations for each recommendation?
  - overall_satisfaction: your overall impression of the system.

Each rating must be an INTEGER from 1 to 5.

3. Provide a 1-3 sentence justification.

Return a single JSON object with keys:
"relevance", "diversity", "personalization", "explanation_quality", "overall_satisfaction", "comment".

Conversation:
---
{conversation_text}
---
"""
```

User Preference: I like action movies.
User ID (press Enter for session-based memory): emily_action

--- Recommendation ---
Based on your long-term preference for action movies and your latest message reiterating that preference, here are the best 5 movie recommendations from the candidates provided:

1. **Momentum (2015)**
   - **Genres**: Action, Crime, Thriller
   - **Overview**: This film features Alex, an infiltration expert entangled in a government conspiracy. It promises intense action and a gripping storyline, making it a perfect fit for action enthusiasts.
   - **Top Cast**: Olga Kurylenko, Morgan Freeman, James Purefoy

2. **Assassins (1995)**
   - **Genres**: Action, Crime, Thriller
   - **Overview**: This movie follows assassin Robert Rath as he navigates the dangerous world of contract killing. With its thrilling plot and action-packed sequences, it aligns well with your love for action films.
   - **Top Cast**: Sylvester Stallone, Antonio Banderas, Julianne Moore

3. **Misfire (2014)**
   - **Genres**: Action, Crime
   - **Overview**: Featuring a DEA agent battling a drug cartel, this film is filled with explosive action and high stakes, appealing directly to your taste for action-packed narratives.

# Qualitative Results - User Studies

10 peers + 3 paid (UserInterviews.com)

n = 13 users

Would watch the recommendation

85% (11/13)

Agreed with rec (for movies already seen)

77% (10/13)

"Because you loved..." helped decision

2% (12/13)

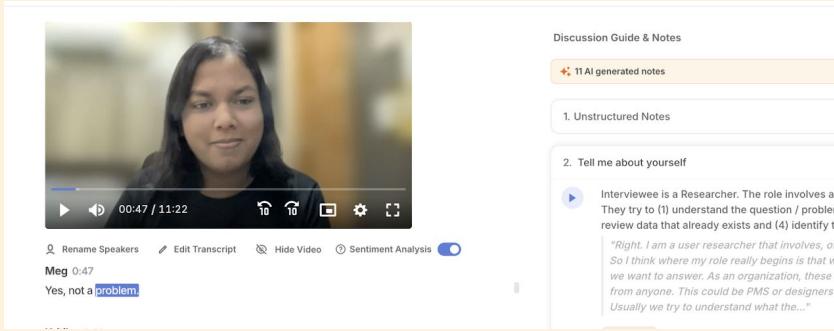
Would trust over Netflix/Spotify recs

62% (8/13)

**Key Finding: Explainability Drives Trust**

92% of participants found explanations helpful — highest rated feature.

Users cited transparency as main differentiator from existing streaming services.




user interviews

https://github.com/niksharma99/COMS6998FinalProject

# Github Repo