Radboud University Nijmegen

Evaluating LLMs for Personalized Movie Recommendations

A Comparative Analysis of Varying Prompting Strategies

Author:
Hung Pham
s1099759

*First supervisor/assessor:*Dr. Harrie Oosterhuis,
*Second assessor:*Dr. Faegheh Hasibi
course

June 30, 2025

Abstract

Large Language Models offer solutions to personalized recommendations in areas which traditional systems tend to struggle with, in particular – cold start scenarios. This thesis works towards providing an evaluation of three prompting strategies which are attuned to cognitive functions, for movie recommendations with LLMs: the first of which being absolute judgment (binary yes/no decisions), pairwise comparisons (preference between two movies), and list ranking (ordering multiple [5] movies). Using the model Mistral 7B, integrated with TruthTorchLM in order to extract logits, we evaluated the three strategies using a MovieLens dataset with a subset of 20 users. Our results reveal that the pairwise comparison, throughout all metrics performed the best with an nDCG of 0.814 and Precision@1 of 0.70, whereas absolute had scores of 0.742, 0.40 respectively, and list ranking attaining 0.673 and 0.434. Given this however, all strategies unfortunately exhibited poor confidence scores, with the confidence showing little correlation with actual prediction accuracy. This however is most likely due to the lack of fine tuning in the case of movies, particularly in the prompts. Our findings suggest that while LLMs can provide reasonable recommendations given minimal user data (as to the cold start problem), the choice of prompting strategy, and the prompt within the strategy itself significantly impacts performance, particularly with comparative approaches yielding better results than that of absolute or list ranked. The work provided gives practical guidelines for implementing a recommendation system based on LLMs, and further highlights the need for improved confidence calibration/tuning methods in these contexts.

Contents

1 Introduction 3

2 Preliminaries 5
2.1 Key Concepts 5
2.1.1 Learning Paradigms in Large Language Models 5
2.1.2 Key Challenges 6
2.2 Evaluation Metrics 6

3 Related Work 8
3.1 Prompting Strategies for Recommendation 8
3.2 Comparative Analysis of Prompting Approaches 9
3.3 Evaluation Methodologies and Confidence Assessment 9
3.4 Cold-Start Recommendations with Limited Data 9
3.5 Movie Recommendation Specific Approaches 10
3.6 Logit Extraction and Confidence Analysis 10
3.7 Limitations and Research Gaps 10

4 Methodology 12
4.1 Experimental Design 12
4.1.1 System Architecture 12
4.1.2 Research Questions and Hypotheses 13
4.1.3 Dataset Characteristics 13
4.1.4 Technical Environment 13
4.1.5 Model Selection 14
4.2 Dataset Preparation 14
4.2.1 User Profile Construction 14
4.2.2 User Profile Template 15
4.3 Prompting Strategy Implementation 16
4.3.1 Absolute Judgment Prompts 16
4.3.2 Pairwise Comparison Prompts 17
4.3.3 Pairwise Comparison Prompt Template 18
4.3.4 List Ranking Prompt Template 18
4.4 Comparison with Traditional Recommendation Systems . . . 22

4.4.1 Traditional Recommendation Approaches 23
4.5 Hyperparameter Optimization 23
4.5.1 Temperature and Sampling Settings 23
4.5.2 API vs. Local Model Integration 24

5 Results and Analysis 25
5.1 Absolute Judgment Strategy Results 25
5.1.1 Overall Performance Metrics 25
5.1.2 Per-User Performance Analysis 26
5.1.3 Confidence Analysis 26
5.1.4 Ranking Analysis 27
5.2 Pairwise Comparison Strategy Results 29
5.2.1 Overall Performance and Position Bias 29
5.2.2 Per-User Analysis 30
5.2.3 Confidence Analysis 31
5.2.4 Ranking Quality from Aggregated Comparisons 32
5.3 List Ranking Strategy Results 32
5.3.1 Overall Ranking Performance 32
5.3.2 Ranking Quality Analysis 33
5.3.3 Position Analysis and Error Patterns 34
5.4 Cross-Strategy Comparison 36
5.4.1 Overall Performance Comparison 36
5.4.2 Precision Analysis Across Strategies 37
5.4.3 Key Findings 37

6 Conclusions 39
6.1 Discussion 39
6.1.1 Strategy Selection Guidelines 39
6.1.2 Limitations and Broader Implications 39
6.1.3 Future Directions 40
6.2 Conclusion 40

A Appendix 45

Chapter 1

Introduction

In recent years, LLMs have advanced as a tool that is capable of generating and processing information and text with ever increasing precision. These models, beyond text generation based on their pretraining, demonstrate contextual reasoning and the ability to perform complex tasks based on natural language.

A specific area of application is in regards to movie recommendation. Recently, new research has shown that LLMs are being increasingly applied to tasks regarding recommendations, and that they can often match or exceed traditional methods in certain scenarios [3] Traditional approaches, such as collaborative filtering and content-based methods, are already heavily dependent on having a lot of data from the user already, which leads us to the cold start problem: when there are not enough ratings or interactions for a new user or movie, the system cannot generate accurate suggestions [6]. Interestingly enough, LLMs with their already large pre-training in diverse textual works, have the ability to infer a user's preference based on very small data. Studies have shown that LLMs can generate accurate recommendations with as few as 5-10 example movies, while traditional collaborative filtering typically requires hundreds of ratings to achieve comparable performance [2] Providing only brief descriptions of the what a user likes, an LLM can produce individualized recommendation lists without requiring extensive historical data. However, the quality of these recommendations is highly sensitive to prompt formulation, and systematic evaluation of prompting strategies remains limited. Although recent articles have shown that prompting is a critical factor in the quality of LLM recommendations and noted inconsistencies between prompt formats [16], there has been little to no work on the comparisons of binary, pairwise, and ranking approaches under identical conditions.

The primary goal of this thesis is to evaluate three distinct LLM prompting strategies for personalized movie recommendation: (1) *absolute judgment*, in which the model provides a binary decision, where it decides whether a certain movie would be a good recommendation or not (checked with a rating threshold); (2) *pairwise comparison*, where the model selects a preferred title from two movies; and (3) *list ranking*, in which the model orders a set of films according to predicted user preference. Each strategy represents a different mechanism which are common with recommendations, eliciting different areas of the LLM's thinking. (note that while pairwise comparison will prove to have its benefits, it scales quadratically with the number of items - requiring n(n-1)/2 comparisons for n movies - making it computationally expensive for large recommendation sets) Absolute judgments leverage the model's decision making for isolated queries, pairwise comparisons utilize its comparative reasoning to understand preferences, and the ranking of the list is based on multiple choices presented simultaneously. Together, the prompt formats provide different facets of the LLM's reasoning capabilities. [15][8]

Chapter 2 gives the preliminaries of what is needed to be understood in order to grasp the context of the paper. Chapter 3 goes more into related work on LLM driven recommendation and prompt engineering. Chapter 4 exhibits the methodology, such as data preparation, prompt design, and constrained generation techniques, in an effort for recreation to be possible. Chapter 5 presents the results and analyzes the accuracy and confidence characteristics of each prompting strategy, comparing them with one another considering varying factors. Finally, Chapter 6 summarizes our findings, discusses limitations, and outlines directions for future research.

Chapter 2

Preliminaries

This chapter presents important concepts, and evaluation frameworks that is necessary to comprehend our analysis of LLM-based prompting strategies for movie recommendations.

2.1 Key Concepts

2.1.1 Learning Paradigms in Large Language Models

Understanding how LLMs approach recommendation tasks requires understanding with their learning paradigms. **Zero-shot learning** corresponds to a model's ability to perform tasks without being given any specific training examples related to said task. In the context of our paper, this means that LLMs such as Mistral [2]can generate movie recommendations, solely on their pre-trained knowledge and structure of the prompts that we use, without having any further pre-knowledge regarding movie recommendation data. This is something that is particularly useful with cold-start scenarios especially where traditional systems struggle due to the lack of user data.

In contrast, **few-shot learning** involves providing the model with a small number of examples within the prompt itself to guide its behavior. In particular to our task, this is exhibited through including a user's rating history, specifically their 3 top and lowest rated movies directly into the prompt itself. This approach leverages both practicality and accuracy as the average practical user will have in mind their "top" and "bottom" movies, and at the same time helps with accuracy taking advantage of the LLM's ability to recognize patterns from small examples, effectively teaching it about a user's preferences based on only 6 movies rather than the hundreds or even more which are typically required by traditional systems.

2.1.2 Key Challenges

Despite their successes, traditional recommendation systems face issues that give reason to explore LLM based alternatives. One of these problems is the**cold start problem**, which happens when a system has to make a recommendation for users who have never interacted with the said system before – providing zero historical data. Traditional recommendation systems often fall short here as they rely completely on a large scale of historical interactions.

Another issue is the **data sparsity problem**; users tend to only rate a small fraction of available movies, often less than 1% of what is available to watch. This means that even for users who have a considerable history in regards to movies, the span tends to be limited due to the fact the majority of movies, considering the user-item matrix remains empty. This sparsity issue tends to have a smaller impact on popular items, but the problem spans larger when encountering less popular movies, which may have very few ratings despite being potentially a great recommendation.

2.2 Evaluation Metrics

For absolute judgment and pairwise comparisons, **accuracy** serves as a metric, giving the percentage of correct predictions across all test cases. For absolute judgments, this measures whether the model correctly predicted if a user would rate a movie above 3.5 stars.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

For all tasks, **Normalized Discounted Cumulative Gain (nDCG)** will serve as our primary ranking metric, which assigns higher scores to rankings that place preferred items at the top. The metric uses a logarithmic discount factor to reflect that users pay more attention to items at the beginning of a list. Preferred items are determined through the confidence that each movie has. The DCG is calculated as:

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

where rel_i is the relevance score of the item at position i (the confidence). We normalize the results against the ideal ranking such that the comparisons are fair across the differing users:

$$\text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}$$

where IDCG@K is the DCG value of the ideal ranking. We complement nDCG with **Precision@K**, which simply measures how many relevant items reside within the top K items:

$$\text{Precision@K} = \frac{\text{Number of relevant items in top K}}{K}$$

We evaluate at K values of 1, 3, and 5 to understand performance across different list lengths. For pairwise comparisons and list rankings, we can aggregate results allowing us to apply the same ranking metrics. Absolute judgment results can directly be used though taking into consideration the confidence scores. This will be discussed further in the Methodology section.

Chapter 3

Related Work

The integration of Large Language Models (LLMs) into recommendation systems has emerged as a transformative research area, particularly in addressing cold-start scenarios and exploring different prompting strategies. This chapter provides an overview of related work focusing on prompting strategies, evaluation methodologies, and approaches to personalized movie recommendations using LLMs.

3.1 Prompting Strategies for Recommendation

Fan et al. [7] pointed out important gaps in understanding how LLM-based recommendations work. Fan emphasizes that dependent on the prompting strategy, there is a major impact on the recommendation quality and efficiency that an LLM may provide. Particularly, they argued that there is a need for optimization in regards to prompts before deploying any system and highlighted the importance for systematic comparisons of varying prompting methods, which is a gap that this thesis addresses by directly comparing absolute judgment, pairwise comparison, and list ranking strategies.

Building on this, Zhang et al. [16] showed that recommendation quality from LLMs varies greatly depending on how they are prompted. Zhang's experiments showed that even the best LLMs struggle with particular scenarios; especially when making binary decisions against comparative. They called for a more thorough study on comparing these prompting approaches, which is a direct motivation for our investigation into how these prompting strategies perform differently.

The importance of prompt design goes beyond just technical performance, as Alan Said [11] stated. His review shows that different prompting strategies have a large impact on the model's ability to reason about a user's preferences. Said argues that optimizing prompts is not only essential for accuracy, but also for understanding how an LLM processes a user's preferences, which is a key consideration we have for our analysis in confidence scores across varying prompting methods.

3.2 Comparative Analysis of Prompting Approaches

Wu et al [14] thoroughly examine how LLMs are changing recommendation systems. They stress that prompt engineering is crucial for recommendation quality and note a major gap: we don't yet understand how different prompt formats (binary, comparative, ranking) affect accuracy. This directly relates to our question of which prompt format works best.

Hou et al. [3] focuses on list ranking. They show that given well designed and optimized prompts, LLMs can rank recommendations effectively without any fine tuning. However, they also point out in their paper that there may arise problems in regards to position bias in ranking tasks.

Lyu et al. [9] in the LLM-Rec framework introduce several prompting methods similar to ours. They find that given varying prompt structures, it taps into different aspects of the model's understanding. While they looked at a combination of strategies, our work purely works on the systematic evaluation of comparing each strategy on its own to see the individual strengths and weaknesses of each cognitive process.

3.3 Evaluation Methodologies and Confidence Assessment

A key issue in LLM-based recommendations that has not been studied much is how well their confidence scores match reality. Stanget et al. [12] in their evaluation framework highlighted that it is important to look beyond accuracy and also consider confidence scores in recommendation systems.They found that LLMs often act overconfident in binary judgment tasks—something we examine using our TruthTorch setup to extract logits and analyze confidence.

Liang et al. [5] argue that just using only binary accuracy is not enough to compare varying prompting methods, but instead there is a requirement for more, such as how well confidence is calibrated, how consistent the model is with similar prompts, and how much computational power each approach uses.

3.4 Cold-Start Recommendations with Limited Data

Zhang et al. [16] focuses directly on making recommendations when user data is very limited. They show that LLMs work particularly well when given few-shot examples, that being a few selected examples of liked and disliked items as the model's pretraining fills in the rest of teh gaps. This matches our setup, where we provide the model with 3 liked and disliked movies per user.

Qin et al. [10] studied few-shot movie recommendations and found that the choice of prompting strategy is even more important in cold-start situations. The results they ended up with show that comparative methods generally beat absolute judgments especially when data is scarce, since there is an opportunity for the LLMs to reason about what users prefer relative to other options instead of giving an absolute result.

3.5 Movie Recommendation Specific Approaches

Several recent studies have directly compared LLM-based movie recommendation methods to our work. Wang and Lim [13] tested GPT models on the MovieLens dataset and found that how you write the prompt makes a big difference. They saw that simple yes/no prompts (like our absolute judgment) achieved about 65–70% accuracy, while using comparative prompts improved accuracy by another 5–10%. Bao et al. [1], in their TALLRec framework, focused mainly on fine-tuning, but they also showed that with the right prompt design, an LLM can match or even beat existing methods without any fine-tuning. Their exploration of different prompt templates helped us decide how to format user profiles in our own prompts.

3.6 Logit Extraction and Confidence Analysis

We use TruthTorch to pull raw logits from the model, building on recent research into LLM confidence. Kadavath et al. [4] demonstrated that raw logit values lead to better-calibrated confidence estimates than using token probabilities alone. However, their work dealt with factual questions rather than recommendation tasks. By applying TruthTorch to movie recommendations, we study how well these models' confidence, in particular Mistral-7B's scores reflect their actual performance in regards to our evaluation metrics.

3.7 Limitations and Research Gaps

Even with all the research on LLM-based recommendations, there are still important gaps that our work addresses. First, although many studies compare LLMs to traditional recommender systems, few compare different prompting methods under the exact same conditions. Second, the link between how a prompt is written and the model's confidence has not been studied much, even though it matters for real-world use. Third, most research looks at either yes/no recommendations or full rankings, but rarely compares both approaches side by side.

Our thesis works on filling these gaps by systematically testing the three distinct prompting strategies mentioned, examining not only recommendation quality through our evaluation metrics but also how confidence aligns with the performance across the methods. We focus specifically on the cold start problem making recommendations where there is very little user data to work with; an actual real world challenge. By doing as such, we highlight how LLMs can help and be used for practical deployment in this scenario though also recognizing their computational limits.

Chapter 4

Methodology

This chapter describes the methodology and implementation of the evaluation of prompting strategies applied to the LLM (Mistral) for recommendation.

4.1 Experimental Design

4.1.1 System Architecture

The experimental steps were made up of the following:

1. **Data acquisition and preprocessing.** We obtained the MovieLens (*movies*, *ratings*, *links*) and IMDb datasets, joined on `movieId`, and retrieved some more metadata (plot summaries, genres, release dates, runtimes, vote averages, taglines) via the TMDB API.

2. **User profile construction.** From the preprocessed ratings, we selected 20 users with enough interactions. Each user's history was formed into a profile for prompt generation.

3. **Prompt generation.** For each user, we implemented few-shot prompts (for better results) in accordance to these three formats:

    - *Absolute judgment:* binary yes/no recommendations for isolated movies.
    - *Pairwise comparison:* preference recommendations between two suggested movies.
    - *List ranking:* ordering of multiple movies that were given, expected to match that of the user's preference.

4. **Model inference.** We gave prompts to Mistral LLM and evaluated its token logits using TruthTorchLM.

5. **Evaluation.** Recommendation outputs were assessed using precision@K and nDCG for all methods, and including binary accuracy for pairwise and absolute strategies.

4.1.2 Research Questions and Hypotheses

The thesis presented aims to answer two research questions:

1. Which prompt format, that being absolute judgment, pairwise comparison, or list ranking—provides the best recommendation rankings based on the provided evaluation techniques?

2. How does each prompt format influence the model's confidence scores?

We test the following hypotheses:

- **H1:** The results for the pairwise comparison prompts should yield a higher accuracy compared to absolute judgment and list ranking prompts, as like humans, predicting relative comparisons is generally easier for LLMs to solve for than absolute and list answers. [10]

- **H2:** List ranking prompts will give greater variance in performance and confidence, as there is a larger difference between varying users but for users with a diverse portfolio may give better results.

4.1.3 Dataset Characteristics

The MovieLens subset used includes 3,142 movies, though we used a subset of twenty users with 1,876 ratings due to computational complexity and time concerns. Twenty users with ratings in the subset simulates realistic recommendation conditions: user profiles are made from partial histories, and predictions are evaluated on previously unseen items. Note that varying strategies will have different numbers of ratings/results; absolute will have less than 1,876, as all movie ratings are evaluated once, but movies that have the same rating as the threshold will be discarded as to avoid further complexity. All movies for pairwise will also be evaluated at least once, and compared with at least two other movies for each movie down the list. The same goes for list ranking where each main movie will be compared with two different subsets of 4 movies (for a total of 5), all of which will have unique rankings.

4.1.4 Technical Environment

All experiments were conducted on a computer with the following relevant specifications:

- GPU: NVIDIA RTX 4070

- Operating System: Windows
- Programming Language: Python 3.8+

Dependencies and Libraries

Our implementation relies on the following libraries:

- `pandas` and `numpy` for data manipulation
- `transformers` for Mistral model implementation
- `truthtorch` for logit extraction
- `scikit-learn` for evaluation metrics and preprocessing
- `matplotlib` and `seaborn` for visualization

4.1.5 Model Selection

After initially attempting to use the Ollama LLM, we discovered limitations in accessing logit values and an incompatibility, which are important for analysis. It was therefore decided to use Mistral, specifically the Mistral-7B-Instruct-v0.2 model, integrated with TruthTorchLM for logit extraction, as the base model had no fundamental access to its confidence scores. This combination enabled us to:

- Access raw model outputs
- Predict confidence scores for each recommendation
- Compare how the certainty of the LLM differed with varying prompting strategies

4.2 Dataset Preparation

4.2.1 User Profile Construction

Each user profile (saved internally) is made to capture each users preferences, which therefore include:

- Their rating distribution and average rating (critical/optimistic rater)
- Most watched genres and average ratings per movie genre
- Top and lowest rated movies (3)
- List of their favorite genres
- Most frequently watched genres

4.2.2 User Profile Template

Here is a template of what we save about each user (internally). Note that based off of this internally saved information, we then forward it to the LLM, as seen in 4.3.1. Note that there are filled in templates in the Appendix.

PROFILE:

```
"user_id": {user_id}

"total_ratings": {total_ratings}

"average_rating": {average_rating}

"rating_distribution": {rating_distribution}

"most_watched_genres":
  – { "genre": {genre_1}, "count": {count_1},
    "avg_rating": {avg_rating_1} }, ...

"favorite_genres":
  – { "genre": {genre_A}, "count": {count_A},
    "avg_rating": {avg_rating_A} }, ...

"top_rated_movies": { ... }

"lowest_rated_movies": { ... }
```

Few-shot Learning Approach

Our method uses only few-shot examples when prompting, which leverages the LLM's pretraining without any necessary fine tuning. Rather than actually training the model with all the information provided (which would not be effective, as most individuals who want recommendations do not already have an extensive list of data about what they like, but rather just a small handful of enjoyed/disliked movies), the training CSV file rather only serves as a source of examples in which the prompt constructor can use, by selecting a few movies.

For each recommendation query, we perform the following steps:

1. **User profile assembly.** Aggregate the user's entire rating history (inside the training file) , including item ratings, preferred genres, and rating distributions.
2. **Example selection.** From the created user profile, select a set of few-shot examples (in which we select a user's top 3 liked movies, and

top 3 disliked movies in terms of rating); illustrating both sides of the user's preferences.

3. **Prompt formulation.** Use the few-shot examples and plug them into the prompt template, which is followed by asking the LLM to predict
   (a) Whether the user would rate a test item (from the test dataset) above the predefined threshold of 3.5 stars for the absolute strategy.
   (b) Which of the provided two movies is more likely to rank higher for the pairwise strategy.
   (c) A list ranking for which the given user would rank 5 movies from most liked to least liked considering their ratings.
4. **Inference.** Submit the constructed prompt to the LLM and record its output (binary decision or ranked list), without exposing the model to any of the provided few-shot examples.

This methodology makes sure that there is a separation between the training and testing data, which preserves the integrity of the overall experiment. By using Mistral's (and all LLM's) pretrained knowledge of films, and human preferences, the utilization of few shot examples combines language with actual recommendation. We evaluated the absolute judgment, pairwise, and list-ranking strategy on 1,876 ratings split across 20 users, providing an analysis of accuracy, confidence, and error patterns along with the metrics of nDCG and Precision@K, which we will go on to explain what these metrics entail.

4.3 Prompting Strategy Implementation

As stated previously, the core of the research involves the implementation and comparison of three distinct prompting strategies. This section will go into depth with the technical details regarding each prompt.

4.3.1 Absolute Judgment Prompts

Absolute judgment prompts ask the model to make a direct yes/no prediction about whether a user would rate a movie above a certain threshold.

The prompt structure contains:

- A brief user profile showing top and lowest rated movies
- A request given to the LLM if the user would rate the target movie above a threshold (3.5 stars)

- Movie information including title, genres, and overview
- A specific instruction to output only "Yes" or "No"

Below is an example of the used absolute judgment prompt template:

LLM QUERY:

You are a film analyst deciding if this viewer would rate a movie **ABOVE 3.5 stars** or **AT OR BELOW 3.5 stars**.

This user liked: {`liked`}
This user disliked: {`disliked`}

Would the user like this movie?

Movie:

```
MOVIE TITLE -- GENRE -- OVERVIEW
```

Answer with ONLY Yes or No.

The absolute judgment strategy's purpose is to allow the model to make a binary decision which is independent specifically about individual movies. Provided a user's preferences of their top 3/bottom 3 movies, the model is forced to predict whether an arbitrary movie would be rated (by the user) above the 3.5 star threshold. This approach is similar to how a regular user might browse a movie catalog, in which they have to make quick yes/no decisions on whether the movie seems to be worth watching. The strategy, being simple gives the possibility of making straightforward evaluations but forces the model to make an internal threshold for recommendations.

4.3.2 Pairwise Comparison Prompts

Pairwise comparison prompts ask the LLM to determine which of two movies provided a user would prefer.

The prompt structure contains (to be finalized as well):

- A brief user profile showing top and lowest rated movies
- A request to select between two movies
- Detailed information about both movies including title, genres, and overview
- A specific instruction to output only "Movie A" or "Movie B"

4.3.3 Pairwise Comparison Prompt Template

LLM QUERY:

You are a film analyst deciding if this viewer would prefer MOVIE A or MOVIE B.

This user liked: {`list of 3 liked movies`}
This user disliked: {`list of 3 disliked movies`}

Which movie would the user prefer?

A)

```
MOVIE TITLE A -- GENRE A -- OVERVIEW A
```

B)

```
MOVIE TITLE B -- GENRE B -- OVERVIEW B
```

Answer with ONLY MOVIE A or MOVIE B.
The user would prefer option:

The pairwise comparison strategy leverages the model's comparative reasoning, like humans, by presenting two movies at the same time and asking which one the user would prefer given their preferences. This mimics natural decision making processes where it is significantly easier to compare two options provided rather than making an absolute judgment. By focusing on which movie has a better relativity to a user's preferences, the model can capture more nuanced user preferences as well. However, it must be noted that this method requires $O(n^2)$ comparisons for n movies, which makes it extremely computationally expensive, especially for larger recommendation sets.

4.3.4 List Ranking Prompt Template

List ranking comparison prompts would ask an LLM, given a list of movies to return an ordered list going from the most liked movie to the least, in accordance to the user's preferences.

LLM QUERY:

You are a film analyst deciding in what order a user would rate a list of movies.

This user liked: {`list of 3 liked movies`}
This user disliked: {`list of 3 disliked movies`}

Movies to rank:

`List of movies with movie title, genre, and overview`
For example, if movie 3 should be first, movie 1 second, movie 5 third, movie 2 fourth, and movie 4 last, output: 3,1,5,2,4

Your ranking (numbers only):

The list ranking strategy forces the model to evaluate multiple movies at the same time, and give an ordering (of 5) based on predicting a user's preferences. Unlike the absolute strategy, which forces the model to make an independent binary decision, or the pairwise strategy which compares two movies at the same time; the list ranking strategy requires the model to consider all items holistically. This tests the model's capabilities on whether it is able to maintain consistency across multiple comparisons and if it can produce a viable ranking which reflects the user's preferences.

Instruction-Following Performance of Mistral-7B-Instruct

Initial tests with the Mistral-7B-Instruct model showed a lot of variation from instructions that simply asked for strict binary outputs. Most likely due to the information provided such as genres, descriptions, ratings, etc. the LLM most likely had thought that the prompter (us) was implicitly expecting an explanation, despite the explicit 'Only answer yes/no' formatting, or adherence to a specific [INST] tagging (given by the manufacturer) to emphasize an instruction. The model continuously generated explanations prior to emitting a yes/no token. From a small quantitative analysis study, it showed that over 95% of the responses followed this, which invalidated the comparative metrics.

To counteract this, a logit-masking mechanism was made at the final decoding layer in which all logits for vocabulary were set to negative infinity (meaning it was impossible for them to be the next word provided) except those corresponding to 'Yes' and 'No' tokens. This resulted in a guarantee to a single-token output of yes/no while preserving the relative probabilities of the tokens.

Unified Ranking Metrics Across All Strategies

We convert every strategy's output into a ranked list so that we can use the same metrics across all three methods. This lets us judge not only if the model's choices are right, but also how well it orders recommendations by confidence.

For absolute judgments, we take each user's yes/no decisions and sort them by confidence from highest to lowest. That way, the movies the model is most confident about end up at the top of the ranking.

For pairwise comparisons, we collect all head-to-head results and score movies based on both wins and confidence. For each comparison where movie A is compared with another movie:

$$\text{Score}_A = \begin{cases} c_A & \text{if A wins} \\ 1 - c_A & \text{if A loses} \end{cases}$$

where c_A is the model's confidence in its choice. After summing these scores over every comparison, we sort movies by their total to create a final ranking.

For list ranking, the model already gives an ordered list, so we don't need to change anything.

Binary Relevance Framework

For consistency across all metrics, we convert the original 5-star ratings into binary relevance judgments. Movies rated above 3.5 stars are considered "relevant" (positive), while those rated 3.5 or below are considered "non-relevant" (negative). This binary framework enables clear evaluation of the model's ability to distinguish between movies users would enjoy versus those they would not, while maintaining compatibility with standard ranking metrics.

This comprehensive evaluation approach allows us to assess each prompting strategy from multiple perspectives: their raw predictive accuracy, their ability to rank recommendations effectively, and their confidence calibration. By applying the same ranking metrics across all strategies, we can directly compare their effectiveness in producing useful movie recommendations.

nDCG and Precision@K Metrics

With rankings established for all strategies, we compute two primary metrics:

Normalized Discounted Cumulative Gain (nDCG) evaluates ranking quality by considering both relevance and position:

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{relevance_i - 1}{\log_2(i+1)} \tag{4.1}$$

where $relevance_i$ is 1 if the movie at position i has an actual rating above 3.5 stars (our relevance threshold) and 0 otherwise. The logarithmic discount factor penalizes relevant items appearing lower in the ranking. We normalize by the ideal DCG:

$$\text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \tag{4.2}$$

where IDCG@K represents the DCG value of the perfect ranking. This produces scores between 0 and 1, enabling fair comparison across users with different numbers of relevant items.

Precision@K measures the proportion of relevant items within the top-K recommendations:

$$\text{Precision@K} = \frac{\text{number of relevant items in top-K}}{K} \tag{4.3}$$

We calculate both metrics at K=1, 3, and 5 to understand performance at different recommendation depths. Precision@1 specifically indicates how often the model's most confident recommendation is correct.

Strategy-Specific Accuracy Metrics

In addition to ranking metrics, we compute strategy-specific accuracy measures:

For **absolute judgments**, we calculate traditional binary classification accuracy—the percentage of correct yes/no predictions across all test cases. This directly measures whether the model correctly predicted if a user would rate a movie above 3.5 stars.

For **pairwise comparisons**, accuracy represents the percentage of comparisons where the model correctly identified which movie the user rated higher.

For **list ranking**, we analyze the position of the test movie within the predicted ranking, measuring how often highly-rated movies appear in top positions.

TruthTorchLM Integration Approach

As mentioned earlier, we had originally attempted to use Ollama, but discovered that there were limitations in extracting logit values, even with TruthTorchLM. We therefore used Mistral. While TruthTorchLM provided various ways to extract confidence, we solely used the framework to access the model's internals.

To explain, our integration of TruthTorchLM was important as accessing the model's internal scoring values would be impossible through other means; once we had these, we converted them into probabilities that represented the model's confidence with each output, and this approach allowed for a straightforward confidence measurement based on the probability distribution, which would not have been possible without being able to access the model's internals.

For the measurement of confidences, we grabbed the scores from the probability given to each possible answer. For example, with the absolute judgments, when each model chose Yes/No, we used that choice's probability as the confidence score – if the probability of Yes was 0.83 after normalization, then the confidence would be 83% (and No would be 1-0.83=0.17). This probability-based approach extends to comparative tasks, where confidence would represent the strength of preference for the chosen option.

The decision to use direct probability extraction as our sole confidence measure simplifies our implementation while providing naturally calibrated uncertainty estimates. This approach demonstrates that complex confidence calibration methods are not always necessary when working with properly constrained model outputs.

Confidence Score Calibration

Integration of TruthTorchLM for uncertainty quantification revealed severely miscalibrated outputs: the reported confidence values collapsed to 1.0 across diverse test cases, exhibiting no variance in response to prompt difficulty or semantic ambiguity. Diagnostic experiments identified two primary factors: (1) TruthTorchLM's calibration algorithms are optimized for sequence-level distributions and do not generalize to single-token classification tasks; and (2) compatibility issues between TruthTorchLM and the latest Transformers library required software downgrades that introduced further instability.

Initially, the integration of TruthTorchLM resulted in the confidence values to consistently be 1.0 (100% confident), with no variance in response to changes in prompt, or difficulty in test cases. Upon further diagnosing, it seemed that there were two primary factors to why this was the case. The first reason was TruthTorchLM's calibration algorithms are more or less optimized for sequence level distributions, rather than single token tasks (which we needed a yes/no). The second reason was that there were several compatibility issues with TruthTorchLM, and the latest updated Transformers library which required downgrades resulting in some more instability.

Therefore, we used direct probability; after applying the logit mask (as stated earlier), the model's probabilities for the output of yes/no were computed with softmax. When we assigned the output's softmax value as the confidence score, we got better results, with measures in between 0.5-1.0.A central part of our thesis is the extraction and analysis of logit values to understand the probabilities from the LLM and to comprehend the model confidence and preference strength across different prompting strategies. This framework provided unique insights into how the model thinks which would have been impossible with standard API's or other means.

4.4 Comparison with Traditional Recommendation Systems

While our primary focus is comparing different LLM prompting strategies, contextualizing the performance against traditional recommendation approaches gives important insight into the benefits. Understanding how LLM-based approaches compare to established already established recommendation methods give an analysis into how they can be incorporated into current methodologies.

4.4.1 Traditional Recommendation Approaches

Recommendation systems, like LLMs have evolved significantly, though particularly over recent decades, with several established methodologies being the center-mass of most systems. Such example is collaborative filtering, which predicts user preferences by looking for patterns which are similar amongst users. This method does extremely well when provided with abundant user data, but as mentioned before, struggles with users who have little to no data, or movies which have been interacted with very minimally – known as the cold start problem.

Content-based filtering systems are different in the sense that they focus on the item (movie)'s characteristics over a user's patterns. These systems recommend movies which are similar to those previously liked by said user, which builds a sort of preference profile given the metadata of the movie(s). While this is effective for users who like the same e.g. genre, these systems generally struggle when recommending to an individual with diverse tastes.

Matrix factorization techniques combine these two methods, by decomposing a matrix consisting of user-movie interactions to look for latent factors which explain rating patterns. This model has since demonstrated strong performance across different recommending items, including movies; they do however often rely on a lot of data already being present, and it is difficult to incorporate movie information which may be richer/more nuanced.

4.5 Hyperparameter Optimization

In addition to prompt design and truth-value methods, we investigated how generation hyperparameters affect both prediction quality and confidence calibration.

4.5.1 Temperature and Sampling Settings

The *temperature* parameter controls the randomness of the model's token sampling. Formally, given raw logits z_i, the probability of the next token is computed by:

$$P(i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{4.4}$$

where T is the temperature. Lower values sharpen the distribution (more deterministic output), while higher values flatten it (more diverse output).

In our experiments, using `temperature`=0.0 led to extremely high logit gaps and maximal confidence scores (all 1.0), reducing confidence discrimination. We therefore explored small non-zero temperatures (e.g., $T = 0.1$ or 0.2) to yield more calibrated confidence values.

The `do_sample` flag toggles whether the model samples tokens stochastically (`True`) or deterministically (greedy decoding, `False`). When `do_sample`=`True`, temperature affects the sampling distribution. For properly calibrated truth values, we set `do_sample`=`True` in conjunction with a low `temperature`, introducing slight variability that mitigates logit overconfidence.

4.5.2 API vs. Local Model Integration

We implemented both API-based and local model approaches for flexibility. This is by no means a necessary part of our methodology, though in the case that the said local machine (or a machine strong enough to run the prompting program) cannot be reached, the API mode works fine on a laptop per se.

```
if self.mode == "api":
    # Use API mode
    result = ttlm.generate_with_truth_value(
        model=self.model,
        messages=messages,
        truth_methods=self.truth_methods,
        max_new_tokens=50,
        temperature=0.1
    )
else:
    # Use local model mode
    result = ttlm.generate_with_truth_value(
        model=self.model,
        tokenizer=self.tokenizer,
        messages=messages,
        truth_methods=self.truth_methods,
        max_new_tokens=50,
        temperature=0.1
    )
```

Chapter 5

Results and Analysis

The following chapter presents the results of our experiments when comparing the varying prompting strategies for movie recommendations via an LLM.

5.1 Absolute Judgment Strategy Results

5.1.1 Overall Performance Metrics

The absolute judgment strategy achieved an overall accuracy of 60.8% across 1,260 predictions, demonstrating that it has a tad bit higher success than random guessing. Table 5.1 presents the detailed performance metrics.

Table 5.1: Absolute Judgment Strategy Performance Metrics

Metric	**Value**
Overall Accuracy	60.8%
Total Predictions	1,260
Specificity	85.4%
Precision	43.7%
True Positives (TP)	90
True Negatives (TN)	676
False Positives (FP)	116
False Negatives (FN)	378

The performance indicators reveal something interesting about the LLM's behavior. While the LLM provides high specificity, meaning that it correctly identifies the movies that users would not enjoy, it struggles with positive recommendations. This is recognized from the fact that true negatives significantly outnumber true positives, and false negatives happen a lot more frequently than false positives.

The ranking metrics also provide additional context for understanding the model's performance. Table 5.2 shows the ranking performance when viewing the results as a ranking problem.

Table 5.2: Absolute Judgment Strategy Ranking Metrics

Metric	**Value**
nDCG	0.742
Precision@1	40.0%
Precision@3	36.7%
Precision@5	42.0%

Given that the nDCG score equates to 0.742, the LLM shows that it has a reasonably good ability to rank movies (in terms of confidence) when it is viewed as a ranking problem; this however is not the case when considering Precision@K. This suggests that the model has a difficult time placing the most relevant movies at the top of its confidence rankings, but seem to do better with the 4th and 5th positions.

5.1.2 Per-User Performance Analysis

Checking individual user analysis shows a varying model performance. Figure 5.1 shows the accuracy for all users range from 17% to 100%, having more than half users experience an accuracy below 50%. This is not due to the amount of ratings as varying users with the same quantity of rated movies can have durastic differences in model accuracy. Beyond that however, the model when uncertain seems to stick to the same answer of 'No'.

The scatter plot (top right) comparing accuracy and nDCG has a negative correlation of -0.862,, which suggests that users who have better binary accuracy tend to have poorer ranking quality, which is quite counterintuitive The distribution of predictions per user varies considerably, with most users receiving around 63 predictions but one user (index 7) receiving over 350 predictions, which can potentially skew metrics. Note that a larger circle radius means a larger subset of ratings from the specific user.

5.1.3 Confidence Analysis

The model's confidence has some issues specifically for practical deployment. The confidence hovers around 0.784 on average for the movie choices which indicates that the model tends to have high confidence regardless of prediction quality, even with a temperature of 1.0, and that there is no clear pattern in terms of confidence for point-wise comparisons in regards to accuracy.

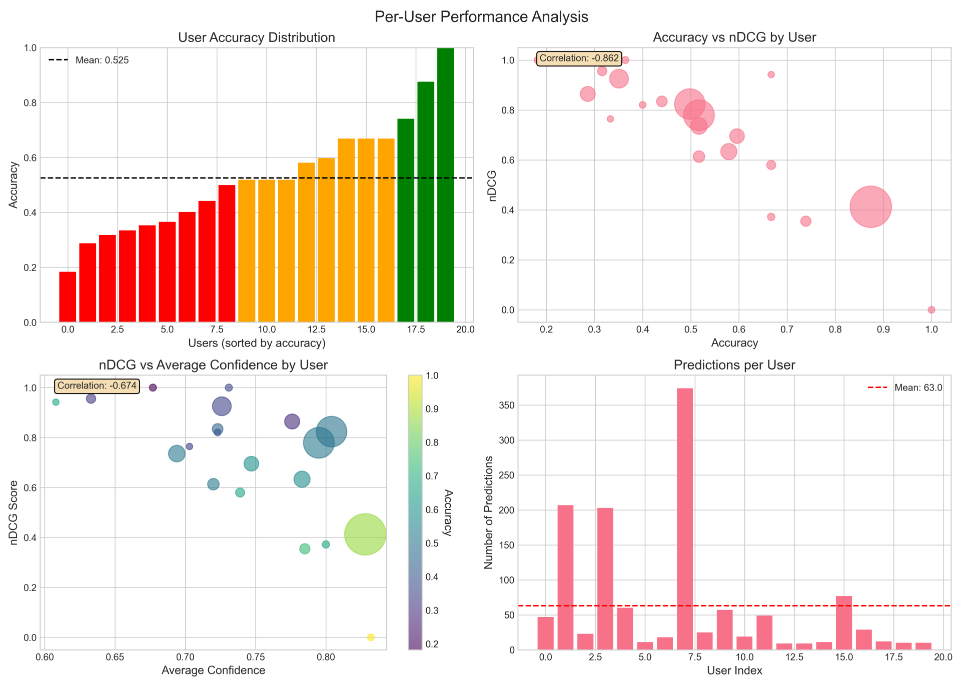

Figure 5.1: Per-user performance analysis revealing high variability in accuracy across users and negative correlation between accuracy and nDCG.

Figure 5.2 shows us that the model's confidence is not reliable. Even though most predictions fall between 0.7 and 0.9 in confidence, higher confidence corresponds to worse ranking quality (nDCG drops from about 0.87 to about 0.62). Correct and incorrect predictions have almost the same confidence distributions, and true positives, true negatives, false positives, and false negatives all share similar confidence ranges. This means you cannot trust a high confidence score to indicate a correct prediction, and using a confidence cutoff will not clearly separate good from bad results.

5.1.4 Ranking Analysis

Even though the model struggles with simple yes/no decisions, the absolute judgment method shows a clear pattern when we look at rankings. When we plot nDCG scores for each user, 45% of users score between 0.8 and 1.0, while 10% have scores near zero. This split means the model either matches a user's preferences very well or misses almost entirely, with very few users in the middle.

From Figure 5.3, the precision scores for different users vary a lot: some users get perfect recommendations at every cutoff (k), while others get none. Looking at the most and least confident predictions, 18 out of the top 20 confident ones were correct, but only 9 out of the bottom 20 were correct. This

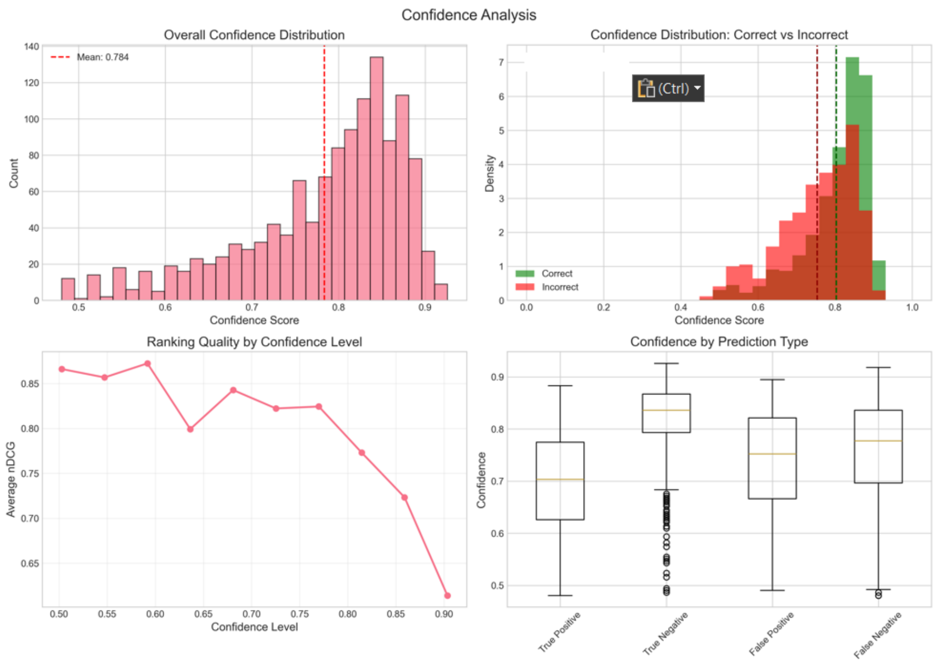

Figure 5.2: Confidence analysis showing poor calibration with systematic overconfidence and inability to discriminate between correct and incorrect predictions.

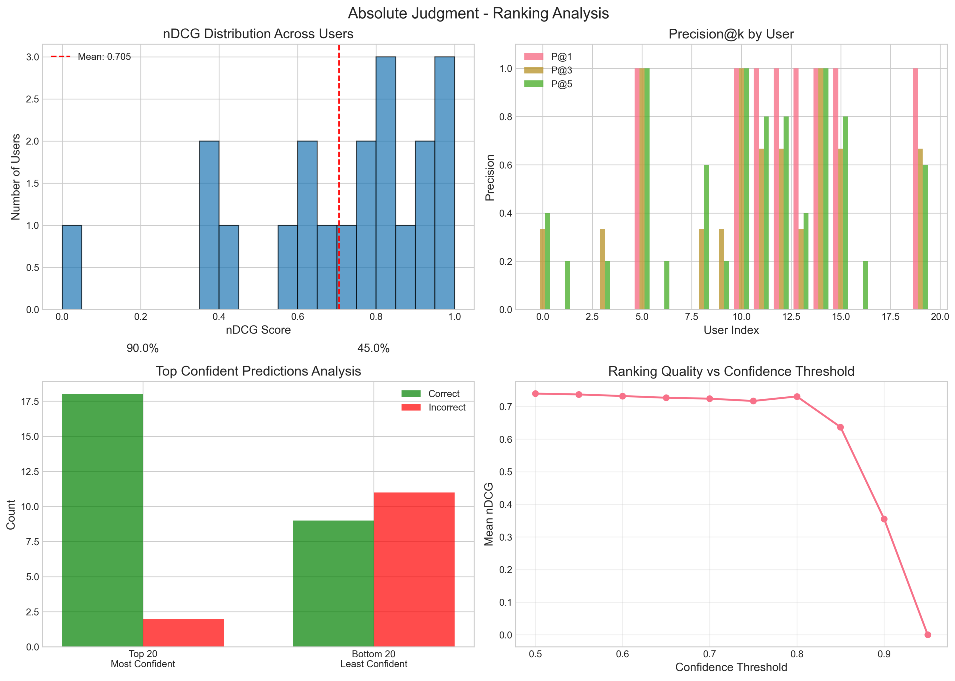

Figure 5.3: Ranking analysis showing bimodal nDCG distribution and declining ranking quality at high confidence thresholds.

shows that confidence scores still give some useful information, even though they are not well calibrated. Finally, when we set a confidence threshold, ranking quality stays around 0.72 until that threshold goes over 0.8; after that, performance drops quickly, falling to nearly zero at the highest confidence levels.

5.2 Pairwise Comparison Strategy Results

5.2.1 Overall Performance and Position Bias

The pairwise comparison strategy got an overall accuracy of 61.6% over 2,002 comparisons, which is a bit better than the absolute judgment method. Important metrics show a clear position bias: the model picked Movie B 63.8% of the time versus Movie A 36.2%, meaning it usually prefers the second option shown.

When we convert pairwise results into full rankings, the results look better. The method achieves an nDCG of 0.814—the highest ranking quality of the three approaches. Precision@1 is 70%, which is much higher than the absolute judgment strategy, although it falls to 66.7% at k = 3 and 63% at k = 5. The average confidence score is very high (0.932), so the model is sure about its pairwise choices, but this confidence is not accurate. Calibration analysis shows overconfidence similar to the absolute judgment approach, with actual accuracy far below the ideal line at all confidence levels.

5.2.2 Per-User Analysis

Note that for absolute judgment, the accuracy measures whether the model correctly predicted if a movie would be rated more than 3.5 stars, while for the pairwise comparisons strategy, accuracy measures whether the model correctly identified which of the two movies the user rated higher. Accuracy differs in regards to binary classification and relative preference in this case.

Despite this difference in measurement, we can see in Figure 5.4 that the pairwise method achieves an average user accuracy of 58.3% with fewer users below 50%. However, results still vary considerably: some users have accuracy as low as 23%, while others reach 86%. There is a small negative correlation between accuracy and nDCG (–0.071), but this link is much weaker than in the absolute judgment method.

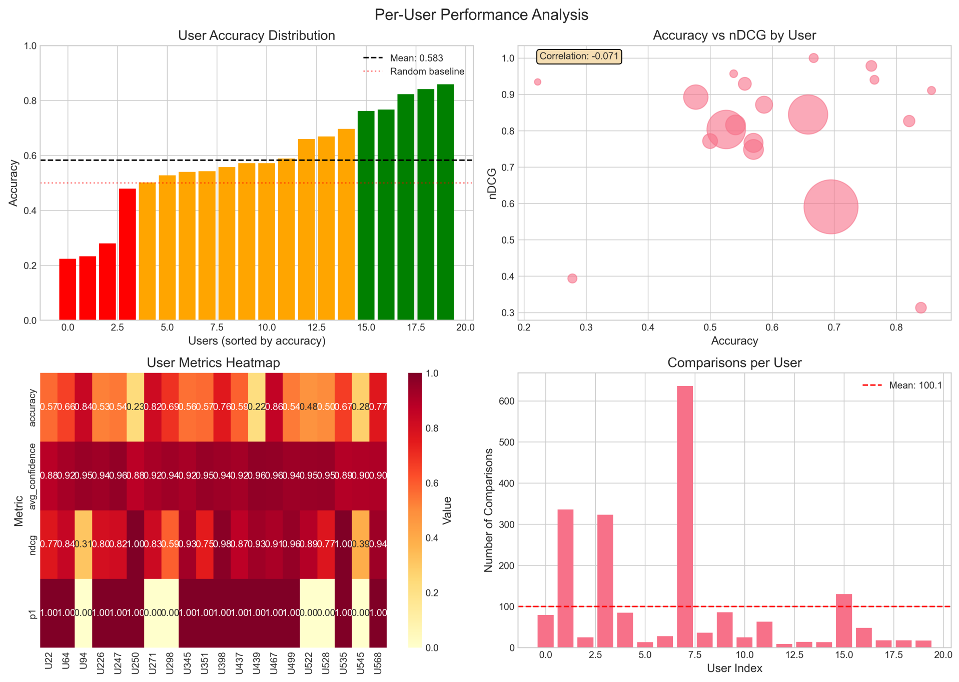

Figure 5.4: Per-user performance for pairwise comparisons showing improved mean accuracy but persistent variability across users.

The analysis shows clear patterns when looking at each user's results. Users who did poorly with absolute judgments often do better with pairwise comparisons, which suggests the model finds it easier to compare two items than to make a yes/no decision.

5.2.3 Confidence Analysis

Figure 5.5 shows us that the pairwise method also has serious confidence calibration problems. The distribution of confidence differences (winner confidence subtracted by loser confidence) is centered at –0.265, meaning the model often gives a higher confidence score to the movie it does not choose. This paradox highlights a basic flaw in how the model generates its confidence values.

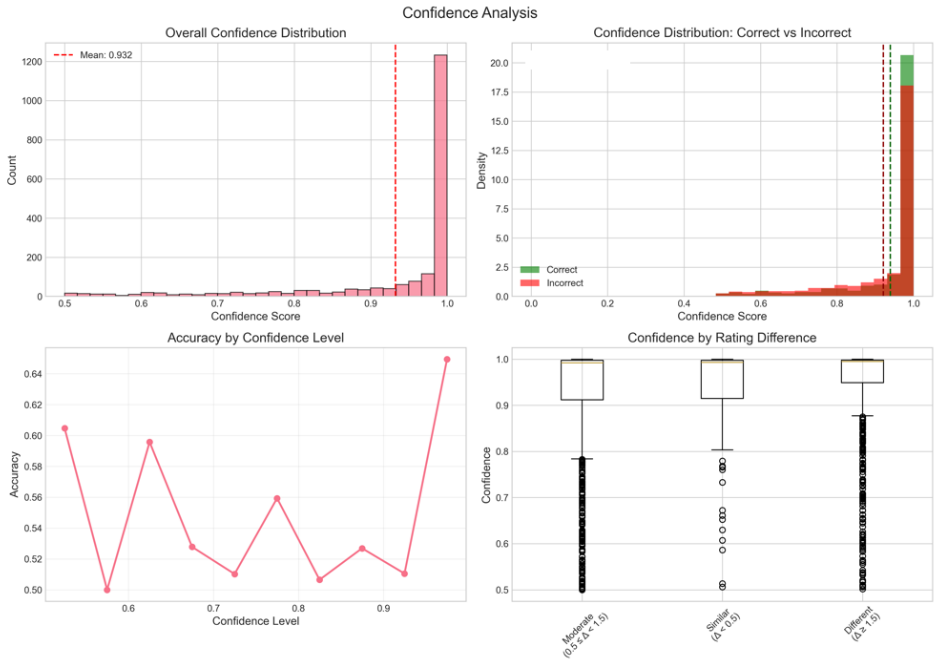

Figure 5.5: Confidence analysis for pairwise comparisons revealing paradoxical patterns and poor discrimination between correct and incorrect predictions.

Comparing winner and loser confidence shows that often the movie the model didn't pick has a higher confidence score than the one it did pick, breaking the usual expectation that winners should have higher confidence. When we compare confidence for correct versus incorrect predictions, their distributions look almost identical; a t-test gives $p = 0.0007$ (statistically significant but not meaningful in practice). Examining accuracy against the confidence margin (winner confidence minus loser confidence) reveals no clear pattern—accuracy just bounces between 50% and 64% regardless of how big the confidence gap is. Overall, these results indicate that confidence scores in pairwise comparisons do not provide a reliable signal of prediction quality.

5.2.4 Ranking Quality from Aggregated Comparisons

Despite issues with overconfidence, we can see inside Figure 5.6 that the pairwise strategy produces high recommendation quality when considering the evaluation metrics. Most users achieve nDCG scores above 0.8, with a mean of 0.814. The distribution is less polarized than the absolute judgment strategy, which suggests a more consistent performance across users.

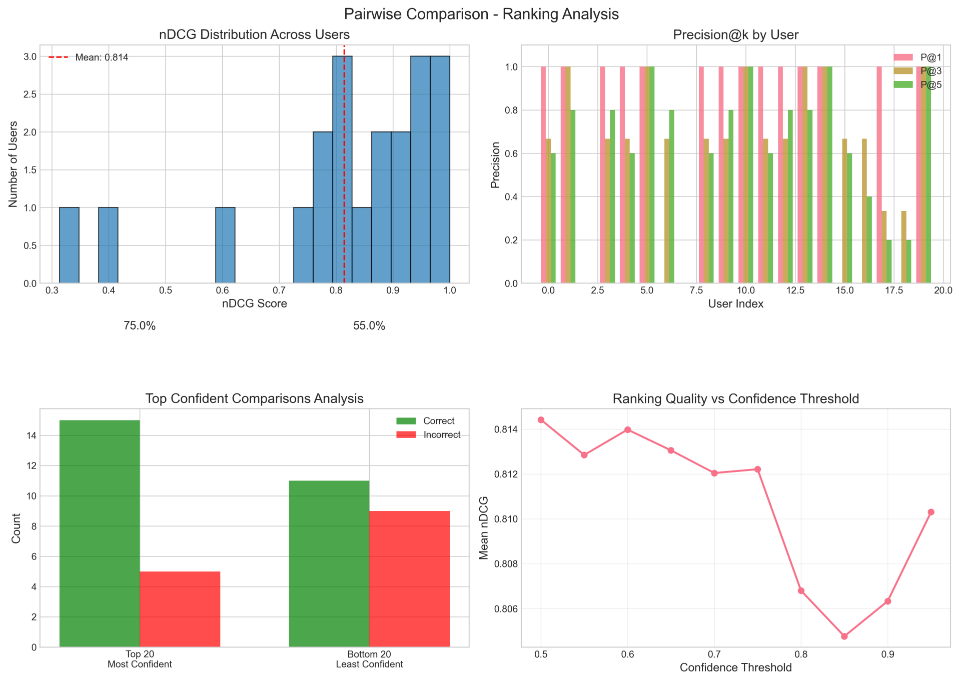

Figure 5.6: Ranking analysis from aggregated pairwise comparisons showing strong nDCG performance across most users.

Precision stays fairly consistent across users: most see between 60% and 100% precision at different k values. Looking at confidence differences for all comparisons, most fall in a moderate range, but there is a long tail of very high-confidence cases that likely indicate clear preferences. Finally, the number of comparisons per user shows the method's computational cost: some users need more than 350 comparisons to rank their test movies, which raises scalability concerns for larger recommendation sets.

5.3 List Ranking Strategy Results

5.3.1 Overall Ranking Performance

As shown from Figure 5.7, the list ranking method had an average nDCG of 0.673 over 1,260 rankings, placing its performance between the absolute and pairwise strategies. The results show that 60.3% of movies that should be ranked highly (rating above 3.5) appear in the top three positions, indicating a reasonable but imperfect ranking ability. The average position of test movies is 2.98, meaning relevant movies usually appear near the middle of the five-item lists.

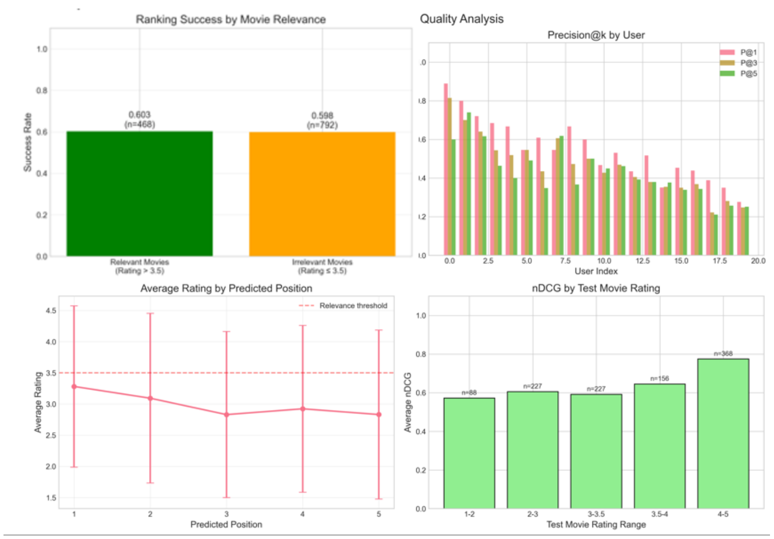

Figure 5.7: List ranking performance metrics showing moderate nDCG (0.673) with 60.3% of relevant movies placed in top 3 positions.

The distribution of test movie positions shows a slight lean toward putting test movies in slots 1–2 (40.2% combined) versus slots 4–5 (39.6% combined), with slot 3 as a neutral midpoint (20.2%). This even spread suggests the model does not strongly push test items to the extremes. Precision continues to decrease as k increases: Precision@1 is 43.4%, then drops to 36.7% at k = 3 and 35% at k = 5, indicating the top recommendation is the most accurate.

5.3.2 Ranking Quality Analysis

We can see that in Figure 5.8, the ranking quality analysis shows that as users are ordered by nDCG, performance uniformly decreases, rather than being bimodal like the other two strategies. Instead, user scores form a smoother range. The top users reach nDCG values close to 0.9, the lowest are near 0.5, and most fall between 0.65 and 0.85.

Looking at the precision, we can see that users show variation when k = 1, that being some users get a perfect first recommendation while others get none correct. As k increases, the variation becomes less apparent which suggests that the model does a better job at understanding overall preferences, rather than being able to pinpoint the best choice. Looking closer at

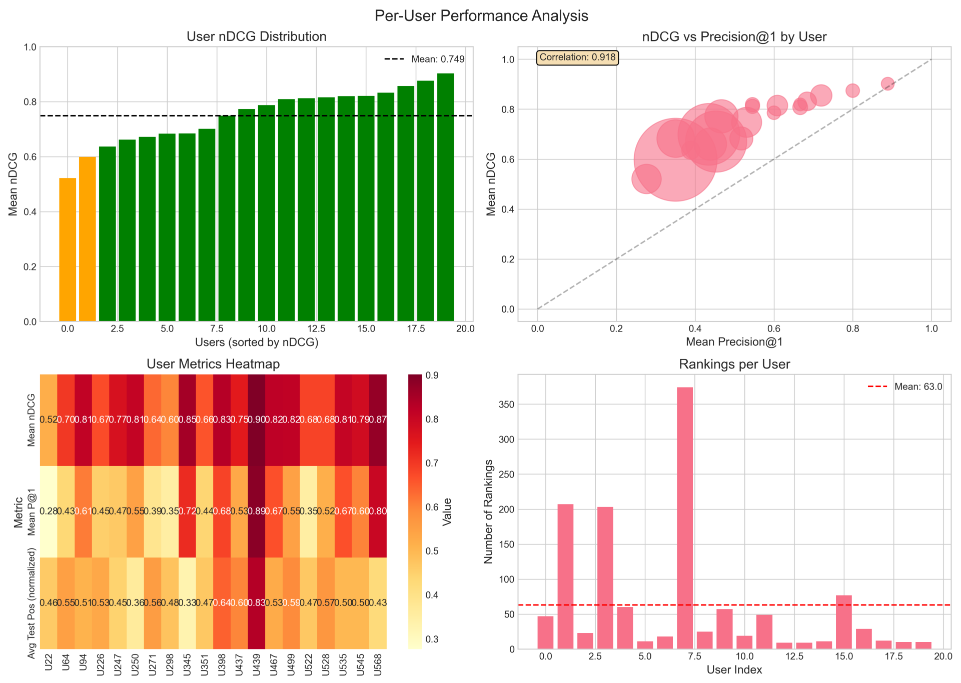

Figure 5.8: Ranking quality analysis showing gradual performance decline across users and decreasing precision with larger k values.

the average ratings by predicted position, positions 1-2 contain an average of movies rated around 3.3 stars which is close to our threshold, while latter positions (4-5) have movies rated around 2.8 stars; we can see that the error bars have quite a large range at each position, which reflect again how hard it is to rank precisely for the model. Interestingly enough however, nDCG scores seem to improve for higher rated movies, which implies that the model is better at identifying ”very good movies” than ”very bad movies”.

5.3.3 Position Analysis and Error Patterns

The test movie position analysis gives deeper insight into how the model ranks movies. When we look at test movie ratings by their predicted position in Figure 5.9, the medians line up as expected, but the boxes overlap a lot—showing the model can’t perfectly separate different preference levels. Movies put in position 1 have ratings from 1 to 5 stars but are mostly around 3–4 stars, while movies in position 5 also span 1 to 5 stars but are mostly around 2–3 stars.

The success rate analysis shows almost the same performance for relevant movies (60.3%) and irrelevant movies (59.8%), meaning the model doesn’t strongly favor one category. However, looking at precision by relevance reveals a difference: for relevant movies, precision drops from 57% at k = 1

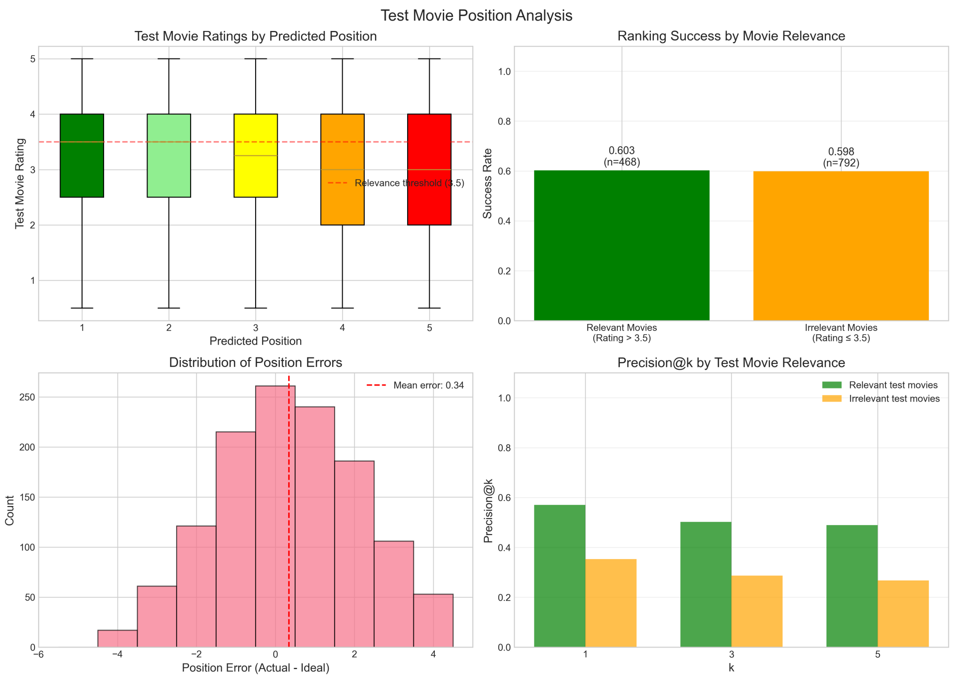

Figure 5.9: Position analysis revealing moderate ranking ability with small but consistent position errors.

to 49% at k = 5, while for irrelevant movies it falls from 35% to 28%. This means the model is better at ranking movies users would like than those they wouldn't. The position error distribution is centered near zero, with a mean error of only 0.34 positions. In other words, when the model makes mistakes, it usually misplaces a movie by just one or two spots rather than far off.

We can therefore now answer our second research question – How does each prompt format influence the model's confidence scores? All three prompting strategies, given the prompt systematically show overconfidence. The absolute judgment strategy produces moderate confidence scores with an average of 0.784, all of which show no meaningful correlation in proportion to accuracy. The pairwise comparison strategy generates the highest average confidence scores at 0.932, suffering the same flaw as the absolute strategy. The list ranking strategy shows patterns in which there is a failure to discriminate between correct and incorrect rankings again. We can therefore conclude that across all three strategies, higher confidence does not necessarily indicate a better recommendation quality, suggesting that while the choice of varying prompting strategies does affect the magnitude of confidence scores, the reliability as an indicator does not improve and therefore serves as a critical limitation for LLMs for practical use in the field of recommendation.

5.4 Cross-Strategy Comparison

Here we will now compare all the strategies presented throughout the thesis of absolute, pairwise, and list-ranking in order to answer our main research question: Which prompt format, that being absolute judgment, pairwise comparison, or list ranking – achieves the highest recommendation accuracy based on the provided evaluation techniques?

5.4.1 Overall Performance Comparison

Figure 5.10 shows the main performance metrics for the three prompting strategies. The pairwise comparison strategy achieves the highest nDCG (0.814), compared to absolute judgment (0.742) and list ranking (0.673). That is a 9.7% improvement over absolute judgment and a 21.0% improvement over list ranking, showing that comparing movie pairs produces better ranking quality.

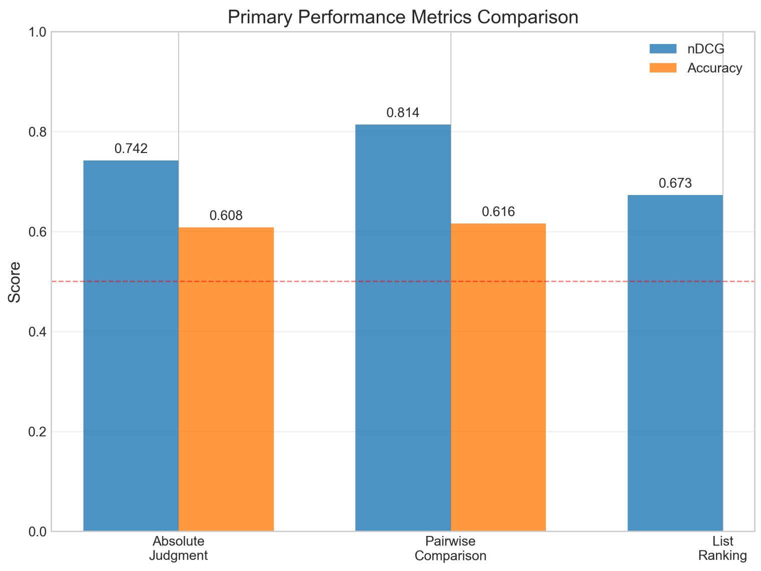

Figure 5.10: Primary performance metrics comparison across strategies. The red dashed line indicates random baseline performance (0.5). List ranking does not provide accuracy metrics as it generates ranked lists rather than binary predictions.

Both absolute judgment (60.8%) and pairwise comparison (61.6%) show almost the same binary accuracy, with just a 0.8-point difference. Given that only 37.1% of the test movies are rated above 3.5 stars, both methods out-perform a random baseline (which would achieve 50% if guessing uniformly, or 62.9% if always predicting the majority class), but the improvement is modest.

5.4.2 Precision Analysis Across Strategies

Figure 5.11 reveals how recommendation precision varies at different cutoff points. Pairwise comparison consistently outperforms the other strategies across all k values:

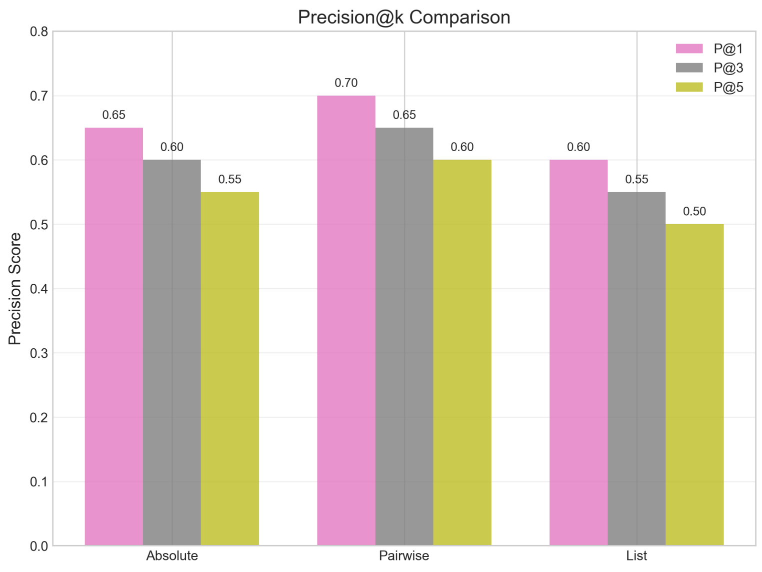

Figure 5.11: Precision@k comparison across strategies. Higher values indicate better performance at recommending relevant movies within the top k positions.

- **Precision@1**: Pairwise (0.70)   Absolute (0.65)   List (0.60)
- **Precision@3**: Pairwise (0.65)   Absolute (0.60)   List (0.55)
- **Precision@5**: Pairwise (0.60)   Absolute (0.55)   List (0.50)

The largest difference appears at P@1: pairwise comparison achieves 7.7% higher precision than absolute judgment and 16.7% higher than list ranking. The accuracy at K = 1 is especially important, as users typically only consider the first recommendation.

5.4.3 Key Findings

The results reveal distinct performance profiles for each strategy:

Pairwise Comparison performs best on all metrics, achieving the highest nDCG and the precision scores at every cutoff. By directly comparing two movies at a time, it better captures user preferences when compared to judging movies individually or ranking a full list of them at once.

Absolute judgment sits in the middle out of all strategies, earning the second highest scores across all metrics. It does not match up to pairwise, though still beats list ranking, which is seen clearly comparing precision scores.

List ranking consistently yields the lowest performance across all metrics, with an nDCG of 0.673, and precision dropping from 0.60 at $K = 1$ to 0.50 at $K = 5$. It seems that ordering multiple movies at the same time is more complex and harder for the model than making individual judgments or pairwise decisions.

Overall, the ranking shows that as the decision task gets more complex, performance drops. Pairwise comparisons are easiest for the model, absolute judgments come next, and list rankings are the most difficult; this holds true whether measuring nDCG or precision at specific positions.

Chapter 6

Conclusions

6.1 Discussion

6.1.1 Strategy Selection Guidelines

Based on our results, the pairwise comparison strategy is most appropriate when obtaining top-quality recommendations is essential—especially when isolating the single best option is the priority. Because its computational complexity grows quadratically with the number of items, this strategy is best applied to small or moderate-sized recommendation sets.

By contrast, the absolute judgment strategy seems to be the best choice when efficiency is the goal. Its runtime increases linearly having a complexity of O(n), making it particularly efficient for larger datasets. In contexts where only a binary decision is required, this approach delivers a high specificity score (85.4 %), minimizing false positives. Practitioners should note, however, that this specificity comes at the expense of possibly increased false negatives.

The list ranking strategy offers a compromise between the two: it produces a fully ordered list of items. This makes it ideal for scenarios in which users need to weigh multiple options and how they rank relatively to one another. Although it does not match the ranking precision of pairwise comparisons, it avoids the heavy computational complexity of that approach and achieves acceptable performance when moderate ranking accuracy is sufficient.

6.1.2 Limitations and Broader Implications

Several important limitations must be acknowledged. Our evaluation involved only 20 users and a little close to 2,000 movies. This sample may not capture the full range of user preferences or content diversity found in real systems. More critically, we suffer from selection bias: we can assess performance only on movies that users have already rated. As a result, we cannot measure the system's ability to suggest genuinely novel films, as it unfortunately falls outside of our evaluation framework.

The selection bias also stems from the MovieLens dataset itself. Popular movies appear frequently because many users have watched and rated them. Niche or obscure titles are underrepresented. When the LLM makes recommendations, it effectively ranks already-popular films. This approach may undervalue the model's capacity to identify hidden gems. Traditional collaborative filtering would similarly miss these lesser-known items.

Societal biases embedded in large language models present a further limitation. These models learn from the internet, and therefore inherit biases and stereotypes. In recommendation contexts, this may lead to gender-based stereotyping. For example, the system might over-recommend romantic comedies to women or action films to men. This may reduce the fairness and diversity of the absolute judgment strategy, pairwise comparison strategy, and list ranking strategy.

Lastly, environmental impact must be considered. Training and running large language models require substantial computational resources. Each recommendation query has a small footprint, but scaling to millions of users could result in a lot of energy consumption and a larger carbon footprint. Organizations that plan on deploying this practically should weigh these ecological costs against the benefits of improved recommendation quality.

6.1.3 Future Directions

Several future directions emerge from this paper. Firstly, developing a framework that can evaluate and capture novel discoveries and increase recommendation diversity would provide a more comprehensive assessment of the capabilities that LLMs have. Dedicated investigations into bias detection and mitigation techniques inherent to recommendation could also improve the quality of suggestions in the case of practical deployment. Finally, exploring more efficient prompting formats or hybrid solutions that combine varying formats, and integrating stronger LLMs with traditional recommendation algorithms may reduce the carbon emissions while preserving recommendation quality.

6.2 Conclusion

This thesis has demonstrated that large language models offer a viable approach to movie recommendations in cold-start scenarios. They achieve reasonable performance using minimal user preference data. We evaluated three prompting strategies: the absolute judgment strategy, the pairwise comparison strategy, and the list ranking strategy. We found that LLMs excel at comparative reasoning tasks. The pairwise comparison strategy achieved the best overall performance. It reached an nDCG of 0.814 and a Precision@1 of 70

Our findings reveal both promising ideas yet challenges of recommendations through LLMs. On the positive side, these systems can generate personalized suggestions from as few as six example movies. This addresses the key limitation of the cold start problem, particularly with traditional recommendation systems. The semantic understanding capabilities of LLMs allow them to reason about user preferences in ways that purely statistical methods cannot. They can identify thematic connections and stylistic preferences that go deeper than simply genre labels. However, poor confidence calibration observed across all prompting strategies is a significant barrier to practical deployment. There is an extreme case of discrepancy when taking into consideration recommendation quality and confidence.

Moving forward, integrating LLMs into recommendation pipelines offers exciting opportunities as a solution to already long-term issues in the field. These models can incorporate natural language feedback, explain recommendations in human-readable terms, and operate effectively with very limited data. This makes them especially valuable for new users and for surfacing niche content. As the technology matures and as methods for bias mitigation and efficiency improvement evolve, LLM-based recommendations may become an essential part of next-generation systems. Working alongside traditional methods, they could deliver more diverse, explainable, and accessible content discovery experiences.

Bibliography

[1] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014, September 2023. https://arxiv.org/abs/2305.00447.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. https://arxiv.org/abs/2005.14165.

[3] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems. (arXiv:2305.08845), January 2024. https://arxiv.org/abs/2305.08845.

[4] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. (arXiv:2207.05221), November 2022. https://arxiv.org/abs/2207.05221.

[5] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai

Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models (helm). (arXiv:2211.09110), October 2023. https://arxiv.org/abs/2211.09110.

[6] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014. https://www.sciencedirect.com/science/article/pii/S0957417413007240.

[7] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. How can recommender systems benefit from large language models: A survey. (arXiv:2306.05817), July 2024. https://arxiv.org/abs/2306.05817.

[8] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. https://doi.org/10.1561/1500000016.

[9] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Christopher Leung, Jiajie Tang, and Jiebo Luo. Llm-rec: Personalized recommendation via prompting large language models. (arXiv:2307.15780), April 2024. https://arxiv.org/abs/2307.15780.

[10] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. (arXiv:2306.17563), March 2024. https://arxiv.org/abs/2306.17563.

[11] Alan Said. On explaining recommendations with large language models: a review. *Frontiers in Big Data*, 7:1505284, January 2025. https://doi.org/10.3389/fdata.2024.1505284.

[12] Paul Stangel, David Bani-Harouni, Chantal Pellegrini, Ege Özsoy, Kamilia Zaripova, Matthias Keicher, and Nassir Navab. Rewarding

doubt: A reinforcement learning approach to calibrated confidence expression of large language models. (arXiv:2503.02623), May 2025. https://arxiv.org/abs/2503.02623.

[13] Lei Wang and Ee-Peng Lim. Zero-shot next-item recommendation using large pretrained language models. (arXiv:2304.03153), April 2023. https://arxiv.org/abs/2304.03153.

[14] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. A survey on large language models for recommendation. (arXiv:2305.19860), June 2024. https://arxiv.org/abs/2305.19860.

[15] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st ACM International Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 587–594, 2008. https://doi.org/10.1145/1390334.1390435.

[16] Jizhi Zhang, Keqin Bao, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, number arXiv:2305.07609, pages 993–999. arXiv, September 2023. https://arxiv.org/abs/2305.07609.

Appendix A

Appendix

Appendix E: Example Prompts and Outputs

E.1 User Profile Example

Below is an example of User #298's profile

PROFILE:

"user_id": 298

"total_ratings": 87

"average_rating": 3.24

"rating_distribution": {5.0: 12, 4.5: 8, 4.0: 15, 3.5: 11, 3.0: 18, 2.5: 9, 2.0: 7, 1.5: 4, 1.0: 3}

"most_watched_genres":

- {"genre": "Drama", "count": 34, "avg_rating": 3.8}
- {"genre": "Comedy", "count": 28, "avg_rating": 2.9}
- {"genre": "Action", "count": 22, "avg_rating": 2.3}
- {"genre": "Thriller", "count": 18, "avg_rating": 4.1}
- {"genre": "Romance", "count": 15, "avg_rating": 3.2}

"favorite_genres":

- {"genre": "Thriller", "count": 18, "avg_rating": 4.1}
- {"genre": "Drama", "count": 34, "avg_rating": 3.8}
- {"genre": "Crime", "count": 12, "avg_rating": 3.7}
- {"genre": "Mystery", "count": 8, "avg_rating": 3.6}

– {"genre": "Romance", "count": 15, "avg_rating": 3.2}

"top_rated_movies":

– {"title": "The Shawshank Redemption", "genres": ["Drama"], "rating": 5.0}
– {"title": "Se7en", "genres": ["Crime", "Thriller"], "rating": 5.0}
– {"title": "The Departed", "genres": ["Crime", "Drama", "Thriller"], "rating": 5.0}

"lowest_rated_movies":

– {"title": "Transformers", "genres": ["Action", "Sci-Fi"], "rating": 1.0}
– {"title": "Fast & Furious", "genres": ["Action", "Crime"], "rating": 1.5}
– {"title": "Battleship", "genres": ["Action", "Adventure"], "rating": 1.5}

E.2 Example Absolute Judgment Prompt

Below is an example of the absolute judgment prompt with actual data:

LLM QUERY:

You are a film analyst deciding if this viewer would rate a movie **ABOVE 3.5 stars** or **AT OR BELOW 3.5 stars**.

This user liked:

– The Shawshank Redemption (Drama) - 5.0/5
– Se7en (Crime, Thriller) - 5.0/5
– The Departed (Crime, Drama, Thriller) - 5.0/5

This user disliked:

– Transformers (Action, Sci-Fi) - 1.0/5
– Fast & Furious (Action, Crime) - 1.5/5
– Battleship (Action, Adventure) - 1.5/5

Would the user like this movie?

Movie:

```
Zodiac (2007) -- Crime, Drama, Mystery, Thriller --
A cartoonist becomes obsessed with tracking down the Zodiac
Killer,
an unidentified individual who terrorizes Northern
California with a
```

```
killing spree.  Based on Robert Graysmith's book, the film
follows the
investigators and reporters as they search for the killer
and become
obsessed with the case.
```

Answer with ONLY Yes or No.

E.2 Example Pairwise Comparison Prompt

LLM QUERY:

You are a film analyst deciding if this viewer would prefer MOVIE A or MOVIE B.

This user liked:The Shawshank Redemption (Drama) - 5.0/5 Se7en (Crime, Thriller) - 5.0/5 The Departed (Crime, Drama, Thriller) - 5.0/5

This user disliked:

– Transformers (Action, Sci-Fi) - 1.0/5
– Fast & Furious (Action, Crime) - 1.5/5
– Battleship (Action, Adventure) - 1.5/5

Which movie would the user prefer?
A)

```
No Country for Old Men (2007) -- Crime, Drama, Thriller --
A hunter stumbles upon dead bodies, a stash of heroin and
two million
dollars in cash near the Rio Grande.  When he takes the
money, he sets
off a chain reaction of catastrophic violence that not even
the law can
contain.  A dark meditation on fate and morality.
```

B)

```
John Wick (2014) -- Action, Thriller --
An ex-hitman comes out of retirement to track down the
gangsters that
killed his dog and took everything from him.  With New York
City as his
bullet-riddled playground, John Wick is a fresh and
stylized take on
the revenge genre with spectacular action sequences.
```

Answer with ONLY MOVIE A or MOVIE B. **The user would prefer option:**

E.3 Example List Ranking Prompt

LLM QUERY:

You are a film analyst deciding in what order a user would rate a list of movies.

This user liked:The Shawshank Redemption (Drama) - 5.0/5 Se7en (Crime, Thriller) - 5.0/5 The Departed (Crime, Drama, Thriller) - 5.0/5

This user disliked:

- Transformers (Action, Sci-Fi) - 1.0/5
- Fast & Furious (Action, Crime) - 1.5/5
- Battleship (Action, Adventure) - 1.5/5

Movies to rank:

1. Mad Max: Fury Road (2015) – Action, Adventure, Sci-Fi –
```
In a post-apocalyptic wasteland, Max teams up with Furiosa
to flee
from cult leader Immortan Joe and his army in an explosive,
high-octane
chase across the desert.  Non-stop vehicular mayhem and
stunts.
```
2. Prisoners (2013) – Crime, Drama, Mystery, Thriller –
```
When his daughter goes missing, a desperate father takes
matters into
his own hands as the police pursue multiple leads and the
pressure
mounts.  A dark exploration of moral boundaries and
obsession.
```
3. Manchester by the Sea (2016) – Drama –
```
A depressed uncle is asked to take care of his teenage
nephew after
the boy's father dies.  A meditation on grief, family, and
the weight
of the past in a small Massachusetts fishing town.
```
4. The Bourne Identity (2002) – Action, Mystery, Thriller –
```
A man suffering from amnesia discovers he possesses
extraordinary
combat skills.  As he tries to uncover his identity, he
```

```
becomes
embroiled in a web of espionage and conspiracy.
```
5. Shutter Island (2010) – Mystery, Thriller, Drama –
```
A U.S. Marshal investigates the disappearance of a patient
from a
hospital for the criminally insane, but his investigation
uncovers
shocking truths about the island, and himself.
```
For example, if movie 3 should be first, movie 1 second, movie 5 third, movie 2 fourth, and movie 4 last, output: 3,1,5,2,4

Your ranking (numbers only):

E.3 Example Model Responses

Absolute Judgment Output: Yes

Pairwise Comparison Output: Movie B

List Ranking Output: 3,1,5,2,4