Recommender Systems Group Project - Report

Group 13, Track I, Topic 2

CHRISTOPH BARTMANN

Large Language Models (LLMs) have been making significant strides across a wide array of applications. In this project, we explore their potential for use in the realm of recommender systems, specifically for news article recommendations. Our system is designed to exploit the capabilities of LLMs in order to generate personalized news article recommendations drawn from the Falter newspaper database. The interaction between the user and the system occurs through a natural language processing interface, with the aim of aligning user queries with news articles that are contextually relevant based on the topics present in both the query and the article. As the system's recommendations are rooted in the plot of the article, we categorize our prototype as a content-based recommender system operating in a session-based manner. The system is designed to accommodate and learn from user feedback in order to enhance its recommendation precision, adjusting its influence adaptively. To round it off, we developed an evaluation pipeline which facilitates hyperparameter tuning, thereby providing avenues for system optimization even in the absence of ground truth values.

ACM Reference Format:
Christoph Bartmann. 2023. Recommender Systems Group Project - Report: Group 13, Track I, Topic 2. 1, 1 (September 2023), 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This project tasked us with leveraging the strengths of Large Language Models (LLMs) like *chat-GPT* to create a news article recommender system. Our assignment involved the development of a prototype that takes natural language user queries and recommends appropriate news articles in response. The ultimate aim is to align user interests with the themes of the news articles. The provided dataset for this project came from the news provider 'Falter' and, while it contained the article's title, text, and other details, it lacked any information about the topics they covered. This presented a distinct challenge. Another difficulty was understanding the user's intended topic from their natural language query. An added complexity was the lack of ground truth values, which makes any evaluation process potentially challenging or extremely time-consuming. This report delineates how we employed LLMs to tackle these problems, describes the workings of the resultant prototype, and discusses its performance according to key metrics.

2 THEORETICAL BACKGROUND

Recommender systems, a subset of information filtering systems, focus on aligning user preferences with a selection of items. They are particularly useful in contexts where a user is presented with a vast array of choices. In the face of an ever-growing digital landscape, recommender systems help users identify the most pertinent items or information by building a model that predicts their likes and dislikes based on their input (both explicit and implicit). They are integral to a variety of digital platforms, including e-commerce, media streaming services, and social media sites. Their

Author's address: Christoph Bartmann, e1616013@student.tuwien.ac.at.

primary function is to personalize a user's online experience. By offering choices that resonate with the user's tastes and interests, recommender systems boost user satisfaction and engagement, and in commercial applications, can stimulate revenue growth by increasing the probability of a purchase or sustained service usage. Recommender systems are regarded as one of the most successful application of machine learning in the industry, contributing significantly to many businesses' success.

In contrast, Large Language Models (LLMs) are transformer-based deep learning models, specifically tailored to handle sequential data such as text. The scaling of model parameters and training data corpus size has enabled these models to exhibit remarkable abilities like complex and knowledge inference, and external robustness [1, 10]. These so-called 'Emergent Abilities' only manifest once a certain threshold of model parameters is attained [8]. For an in-depth understanding of LLMs and their inner workings, we refer readers to the survey by Zhao et al. [10]. This source also offers valuable guidance on prompt engineering across different domains.

Since the API access to sophisticated models such as chat-GPT or GPT-4 was made available, LLMs have been attracting attention across various fields [9]. The field of recommender systems is no exception, with new approaches emerging daily. For instance, Friedman et al. discuss ways to use LLMs in building conversational recommender systems ([2]). This approach closely mirrors the one we adopted in our project, but this source was discovered late in our project timeline and thus did not inspire our work. Other resources, such as the survey by Liu et al. [5], provide a comprehensive overview of the potential that LLMs hold in this field. Lastly, the deployment of chat-GPT plugins, like those offered by *Kayak* or *Expedia*, most likely represents one of the most significant real-world applications of LLM-powered recommendation systems to date [6].

3 METHODOLOGY & APPROACH

In our methodology, we adopted the LangChain framework, which offers a streamlined platform for crafting custom LLMs equipped with several crucial functionalities [3]. Although the framework's documentation lacks explicit instructions on using it for text corpus recommendations, we reasoned that the conventional Q&A LLM approach wouldn't suit our needs. This conventional approach would involve chunking each article, generating its embeddings, and storing them in a vector store. The LLM would then respond to user queries based on the similarity between their input and these embeddings. However, this method wouldn't accurately capture the topics a user is interested in or the topics an article addresses. Hence, we devised an alternate strategy, which we outline in this section and is schematically represented in Figure 1.

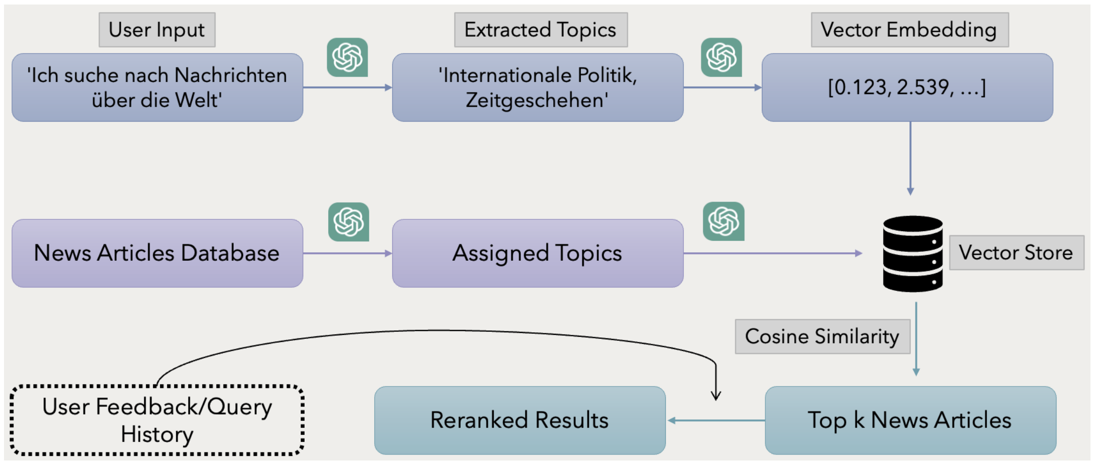


Fig. 1. Workflow of the developed Recommender System for News Articles based on topics

We didn't utilize the entire dataset provided but limited our scope to articles containing less than 5000 characters of text. This decision was made to meet the token restrictions of the LLMs and to adhere to our budgetary constraints. This selection left us with approximately 87% of the original data (5150 articles). We will not delve into the specifics of the dataset here, given that it was employed by a considerable number of other groups. It's safe to presume that the dataset's overview has been sufficiently discussed elsewhere. The key point is that we used 5150 articles, all of which include a title and text. Some other attributes may be missing values, but this does not impact our methodology. As for our LLM, we utilized OpenAI's *gpt-3.5-turbo* model with suitable temperature settings, and for embeddings, we deployed the *text-embedding-ada-002* model with default settings.

3.1 Topic Assignment

As stated earlier, the dataset lacks information on the topics each article covers. To rectify this, we began by curating a list of 40 wide-ranging topics. We then designed an LLM chain to assign a list of topics to each article based on its content, with all available information about the news article provided to the LLM. The designated topics were then embedded and stored in a vector store. This step necessitated several prompt iterations and the production of ten hand-annotated article assignments, one of which was used as a zero-shot example in the LLM chain. These manually labeled examples helped confirm the accuracy of the developed chain.

Upon examining the vector store's contents after embedding the entire dataset, it became apparent that the LLM did not strictly adhere to the predefined list of 40 topics. Instead, it assigned a total of 296 unique topics. This isn't necessarily problematic as it allows for a more detailed breakdown of each news article's topics. Moreover, we observed that a significant number of the 296 topics were seldom assigned, with only the top 15 topics being attributed to more than 5% of the articles. Nearly 96% of the articles were assigned between 2 and 4 topics.

3.2 User Query Handling

Having resolved the issue of assigning topics to articles, we then devised a similar strategy for managing user queries. We created an LLM chain that processes a user query, then assigns it topics drawn from the set available in the vector store. After embedding the resulting list of topics, we conducted a simple cosine similarity search in the vector database, subsequently outputting the top k similar results alongside their respective similarity scores.

Like in topic assignment, we incorporated zero-shot examples in the query, which served to enhance the LLM's grasp of the task and the desired output. To examine the performance of the developed chain, we used the general prompt set intended for system evaluation (see Section 4.3). With this limited query set, we noted that the LLM invented new topics instead of solely selecting from the existing set, which could potentially be disadvantageous. As such, future iterations should implement stricter system prompts.

3.3 Refinement of Preliminary Results

Upon extracting the topics from the user query and conducting the similarity search, we obtain the top k similar articles along with their corresponding similarity scores. However, these are preliminary results, not final recommendations. We adjust the score of each article via two separate mechanisms:

(1) Random Noise: To promote variety in recommendations, we add random noise (derived from a normal distribution with an adaptive standard deviation setting) to the similarity score.
(2) User Feedback: We factor in user feedback collected through ratings as weights for each topic. As not all weights convey the same amount of information, we first compute the weighted average weight, with the weights being the frequency of user feedback on an item discussing the topic. We use a temperature parameter to adaptively adjust the weight influence through multiplication. Other factors that could influence the re-ranking process include the frequency of the topic's appearance in the database.

The impact of these re-ranking procedures adaptively changes with the number of user queries. As we assume our understanding of user preferences improves over time, the weight influence gradually increases, while the randomness decreases. The adaptive schedule for these two modifications follows the two formulas illustrated in Figure 2 and demonstrated in Figure 3.

$$t_{ad} = t_{\max} \cdot \left(1 - \frac{1}{1 + p \cdot n}\right) \quad \text{(1a)}$$

$$r_{ad} = \frac{r_{\text{start}}}{\log(n+1)} \quad \text{(1b)}$$

t_{ad}	The adjusted temperature
$t_{\mathbf{max}}$	The maximum temperature
p	Rate to adapt the change in values
n	The number of queries
r_{ad}	The adjusted noise
$r_{\mathbf{start}}$	The starting noise

Fig. 2. Formulas of the Adaptive Schedules

Further improvements to our method could involve implementing a decay rate, which simulates short-term memory by causing the ratings for topics that have not been rated for an extended period to gradually revert to the default value. Another possibility could be to base the schedule on the number of rated items, rather than the total number of user

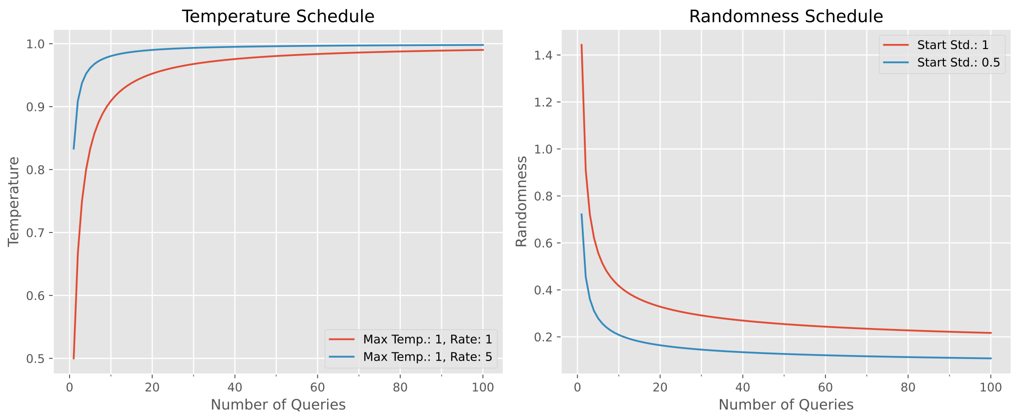


Fig. 3. Examples of the two schedules for the Temperature and Randomness

queries.

It's also crucial to explain how we acquire the weights for each topic. At the beginning of each session, all topics are assigned the same default weights. Users have the opportunity to rate an article as *good*, *bad*, or *ok*. While *good* and *bad* ratings have obvious impacts, *ok* ratings require more careful consideration. We can design them to have no impact, or to bias slightly towards good or bad. A 'severity parameter' determines the effect of a good or bad rating.

Furthermore, we confine the weight values to a specific range. This method ensures diversity and continuous exploration of news articles, even if a particular topic might have received poor ratings in the past. By preventing a weight from reaching zero, we ensure that articles which may be of interest to the user, but involve poorly-rated topics, can still be recommended. Similarly, we impose a maximum limit on the weight to prevent any single topic from overshadowing all others. This ensures that recommendations remain diverse and not too narrowly focused on a single topic.

3.4 Evaluation

Following the implementation of our recommender system, a significant challenge surfaced - how to evaluate its performance without any ground truth data. The ideal solution would be to conduct user studies, but that was not feasible within the scope of this project. As a result, we turned to Language Model (LLM) based evaluation, as recommended by both LangChain's documentation[4] and OpenAI's evals repository[7].

In this arrangement, the evaluation is driven by the LLM. Our pipeline for this process comprises two components. Firstly, we created an LLM chain that generated 100 prompts derived from 100 randomly selected news articles. Each article produced two types of prompts:

- General Prompts: These prompts capture a high-level understanding of the article.
- Specific Prompts: These prompts encapsulate specific details contained within the article.

We then leveraged these prompts to generate recommendations through our pipeline. A simplistic approach to evaluation could be to check if the article ID from which the prompt originated appears amongst the recommended articles. However, we acknowledged the limitations of such an approach, considering the possible relevance of other articles to the prompt, especially for general prompts.

To address this, we employed an LLM to evaluate the relevance of the recommendations. We crafted two different evaluation settings, each with suitable system and human prompts, alongside zero-shot examples:

- Relaxed Evaluation: In this setting, the LLM uses lenient criteria in assessing the correlation between the user query topics and the article topics.
- Strict Evaluation: Conversely, in this setting, the LLM utilizes rigorous criteria for the same assessment.

Our devised approach enables the utilization of generated prompts and facilitates the evaluation of recommendations based on key metrics such as top k recall, precision, and mean average precision (mAP). It also allows us to check whether the original article from which the prompt was derived appears in the recommendations.

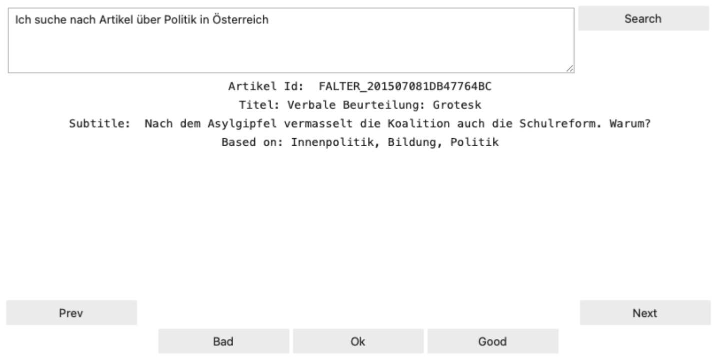
The evaluation chains we developed were further validated on the general and specific prompts sets, using the news article from which the prompt was derived. The general queries yielded a relevance score of 88 and 99 on the strict and relaxed evaluation systems respectively. The specific queries followed suit, with relevance scores of 97 and 98, respectively. The outcomes suggest that our approach offers a promising alternative to user studies for the evaluation of recommender systems.

4 RESULTS

4.1 User Interface

A significant outcome of our project is the development of a user-friendly interface, where users can input any query and receive recommendations for relevant news articles. The presented results include crucial information such as the Article Id, title, subtitle, and the topics the article covers. Once the results are displayed, users can rate them as *bad*, *ok*, or *good*. An example of the user interface is shown in Figure 4. Each time the system is accessed, the weights are reset to their default value, ensuring no information is carried over between sessions.

The prototype we developed has numerous parameters that could potentially influence its performance. We explore a few of these in the following discussion. However, due to the constraints of our budget and our reliance on the LLM chain for evaluation, we primarily focused on the influence of noise and user feedback.



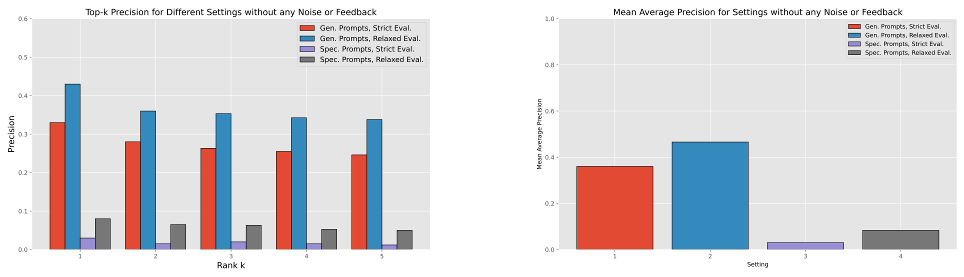
Ich suche nach Artikel über Politik in Österreich

```
Artikel Id:  FALTER_201507081DB47764BC
Titel: Verbale Beurteilung: Grotesk
Subtitle:  Nach dem Asylgipfel vermasselt die Koalition auch die Schulreform. Warum?
Based on: Innenpolitik, Bildung, Politik
```

Fig. 4. Developed User Interface for the Recommender System

4.2 Performance on General and Specific Prompts without Refinement

Before analyzing the influence of any hyper-parameters, we first investigate the out-of-the-box performance of the system without any reranking. We do this by evaluating the performance of the general and specific prompts using both the strict and relaxed evaluation methods, generating five recommendations. The resulting mean average precision is depicted in Figure 5.



(a) Precision of General and Specific prompts on both relaxed and strict Evaluation Chains

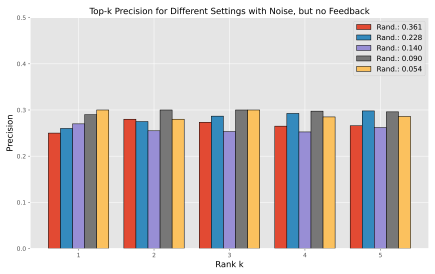
(b) mAP of General and Specific prompts on both relaxed and strict Evaluation Chains

Fig. 5. Performance of general and specific prompts without any Noise or User Feedback

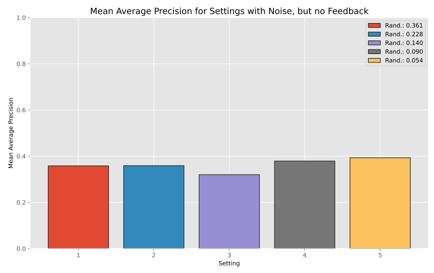
Unsurprisingly, the strict evaluation chain results in lower metrics overall. Comparing the results of the general and specific prompts also delivers the expected outcome. Due to limitations in our budget and the scope of this report, in the following sections we will only report results using the general prompts and the relaxed evaluation chain, as this combination provides the most informative results.

4.3 Influence of Noise

To examine how the noise parameter affects the performance of the system, we set a fixed starting standard deviation of 0.25 and iterated over five different round settings: 1, 2, 5, 15, and 100 respectively. We maintained the number of final recommendations at 5, while the number of initial articles is 25. While we understand that our noise addition is a stochastic process and thus the experiment should be repeated multiple times, we only conducted this experiment once due to resource constraints. Selected results are shown in Figure 6.



(a) Precision of the General Prompts on the relaxed Evaluation Chain with Noise and no User Feedback

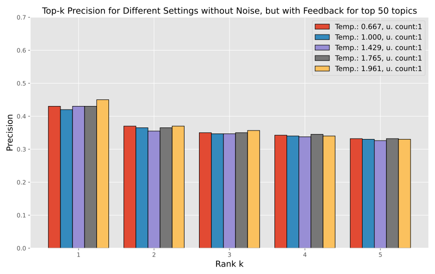


(b) mAP of the General Prompts on the relaxed Evaluation Chain with Noise and no User Feedback

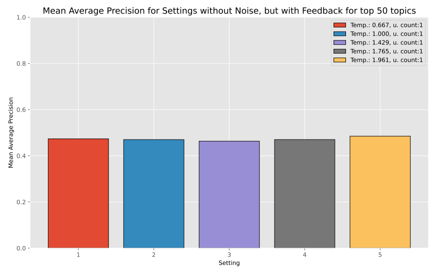
Fig. 6. Performance of General Prompts with Noise, but no User Feedback

4.4 Influence of User Feedback

Investigating the influence of user feedback is more complex than analyzing the impact of noise. To do so, we again utilized the extraction LLM chain to extract the top 20 and 50 topics, respectively, from the general prompts. These topics were assigned a weight of 1, with a user count of 1 or 25, respectively. All other weights were set to a default value of 0.5. From this, we filtered out the 25 most similar articles, which were then reranked. This reranking was performed according to various temperature settings. In this section, we present the results from the setting with the top 50 topics and a user count of 1. The results are shown in Figure 7. The outcomes of all other mentioned settings are provided in the appendix.



(a) Precision of the General Prompts on the relaxed Evaluation Chain with no Noise and User Feedback for the top 50 topics and a user count of 1

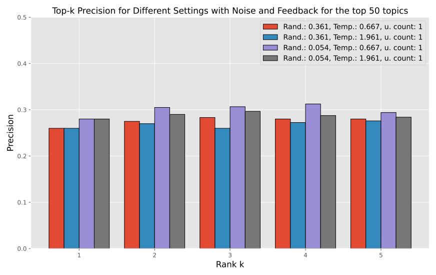


(b) mAP of the General Prompts on the relaxed Evaluation Chain with no Noise and User Feedback for the top 50 topics and a user count of 1

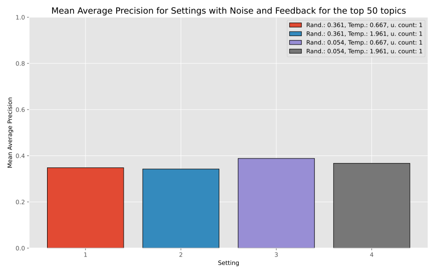
Fig. 7. Performance of General Prompts with no Noise, but User Feedback for the top 50 Topics and a user count of 1

4.5 Influence of Noise and User Feedback

Finally, we explored the combined impact of noise and user feedback on the quality of recommendations. For this analysis, we used two temperature values and two randomness values respectively. We again selected the top 50 topics and a user count of 1. As with previous tests, the top 25 most similar news articles were reranked. The results are presented in Figure 8.



(a) Precision of the General Prompts on the relaxed Evaluation Chain with Noise and User Feedback for the top 50 topics and a user count of 1



(b) mAP of the General Prompts on the relaxed Evaluation Chain with Noise and User Feedback for the top 50 topics and a user count of 1

Fig. 8. Performance of General Prompts with Noise and User Feedback for the top 50 Topics and a user count of 1

5 DISCUSSION

The performance results of our experiments are presented in the previous section, with raw values available in the Appendix. Several insights can be drawn from these experiments.

One observation is that the introduction of noise in the reranking procedure only marginally alters the system's performance. It appears that lower noise values are generally better than higher ones. However, extreme values (either low or high) seem to fare better than intermediate values. It's important to note, though, that these differences are slight, and the experiment should be repeated multiple times to draw more confident conclusions.

As for user feedback, our experiments indicate that a higher temperature setting could be advantageous, especially in earlier ranks, as measured by the top k precision. While we did not present these results graphically, our findings suggest that a user count of 25 is less beneficial than a user count of 1. This is not entirely surprising, as higher user count weights would dominate the similarity score. Given that the general prompts cover a broad variety of topics, this could lead to less accurate recommendations. Along similar lines, models that adjusted weights for the top 50 topics outperformed those that only modified the top 20.

To provide a more comprehensive comparison of performance, we also assessed the model's predictions for randomly selected articles corresponding to each of the general prompts. Figure 9 illustrates the mean average precision (mAP) for all the best-performing settings.

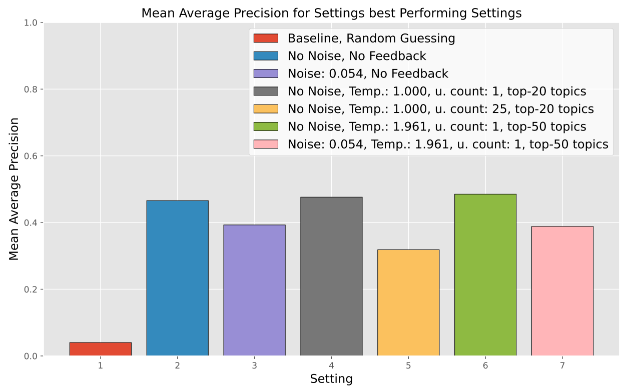


Fig. 9. mAP of the best performing Models on each Setting

As seen in Figure 9, all of our settings significantly outperform a randomly guessing baseline. We also find that the base model, without any post-processing, already performs quite well. Although the introduction of noise seems to slightly decrease the performance, it's worth noting that the goal of introducing noise was to preserve diversity in the recommendations, not necessarily to increase performance. As such, mAP may not be the most suitable metric in this context.

While our findings suggest that a higher user count isn't especially beneficial in this setup, the addition of user feedback at a suitable temperature setting does seem to slightly improve performance.

Finally, it is important to emphasize that the conclusions discussed here are contingent upon the proper functioning of the evaluation chain. While we used the generated prompts to validate its performance, it is still recommended to perform a qualitative evaluation of the system to confirm these findings.

6 CONCLUSION & LIMITATIONS

In summary, this project demonstrates how the capabilities of a large language model (LLM) can be harnessed to construct a recommender system. Given the nature of the task and the data available, alternative approaches would have been significantly more time-intensive and potentially limited in terms of user interaction via natural language. In this project, we developed a content-based recommender system that incorporates user feedback to refine its recommendations over time. Alongside the implementation of a user interface for interaction with the developed prototype, we also designed an evaluation procedure leveraging the capabilities of a LLM. This evaluation methodology facilitated our analysis of the impact of noise and user feedback on model performance, even in the absence of ground truth values.

However, it's important to note that while the project underscores the versatility and power of LLMs in tackling problems where traditional approaches would be more time-intensive or even impractical, it also highlights the need for new skills in model development. The importance of iterating over the used prompts cannot be overstated. Additionally, as even testing and development incurs costs due to the API calls, we were constrained in the range of approaches we could explore. Moreover, while a user has some control over the model's behavior through prompt selection, the system essentially operates as a black box.

Finally, it is worth mentioning that LangChain offers an expansive range of built-in functionalities, such as the incorporation of a knowledge graph into a LLM chain or pre-constructed Retriever chains. With these continually expanding possibilities, the development of recommender systems with LLMs will likely become increasingly straightforward in the future. At this point in time, there is no evidence that LLMs will not play a pivotal role in the future of recommender systems.

REFERENCES

[1] Aakanksha Chowdhery et al. “Palm: Scaling language modeling with pathways”. In: *arXiv preprint arXiv:2204.02311* (2022).

[2] Luke Friedman et al. “Leveraging Large Language Models in Conversational Recommender Systems”. In: *arXiv preprint arXiv:2305.07961* (2023).

[3] LangChain. *LangChain Python Documentation*. 2023. URL: https://python.langchain.com/docs/get_started/introduction.html (visited on 07/05/2022).

[4] LangChain. *LangChain Use Cases - Evaluation*. 2023. URL: https://docs.langchain.com/docs/use-cases/evaluation (visited on 07/05/2022).

[5] Peng Liu, Lemei Zhang, and Jon Atle Gulla. “Pre-train, prompt and recommendation: A comprehensive survey of language modelling paradigm adaptations in recommender systems”. In: *arXiv preprint arXiv:2302.03735* (2023).

[6] OpenAI. *ChatGPT Plugins*. 2023. URL: https://openai.com/blog/chatgpt-plugins (visited on 07/05/2022).

[7] OpenAI. *OpenAI Evals GitHub Repository*. https://github.com/openai/evals. Accessed: 2022-07-05. 2023.

[8] Jason Wei et al. “Emergent abilities of large language models”. In: *arXiv preprint arXiv:2206.07682* (2022).

[9] Tianyu Wu et al. “A brief overview of ChatGPT: The history, status quo and potential future development”. In: *IEEE/CAA Journal of Automatica Sinica* 10.5 (2023), pp. 1122–1136.

[10] Wayne Xin Zhao et al. “A survey of large language models”. In: *arXiv preprint arXiv:2303.18223* (2023).

7 ORGANIZATION

List of Tasks:

- Proposal: Antal Spilyka
- Conceptual Design: Antal Spilyka, Maximilian Tschoner, Max Auernig, Christoph Bartmann
- Implementation: Maximilian Tschoner, Max Auernig, Christoph Bartmann
  - Topic Assignment & Vector Store: Antal Spilyka, Luka Trailovic, Christoph Bartmann
  - User Interface: Maximilian Tschoner, Max Auernig, Christoph Bartmann
  - Refinement Mechanisms: Christoph Bartmann
  - Evaluation: Antal Spilyka, Luka Trailovic, Christoph Bartmann
  - Experimental Testing & Analysis: Christoph Bartmann
  - Code Review & Documentation: Antal Spilyka, Luka Trailovic, CHristoph Bartmann
- Presentation: Antal Spilyka, Luka Trailovic, Maximilian Tschoner, Max Auernig, Christoph Bartmann
- Report: Christoph Bartmann, Antal Spilyka

A APPENDIX RAW DATA

Unless otherwise noted all results were obtained by the following setting: General Prompts set, Relaxed Evaluation chain, default weights of 0.5, initial selected top 25 most similar topics and 5 news articles finally recommended.

Table 1. Experimental Results without any Noise or User Feedback

Prompts	Eval. Type	Randomness	Temperature	P@1/R@1	P@2/R@2	P@3/R@3	P@4/R@4	P@5/R@5	mAP
General	Strict	0	0	0.330/0.160	0.280/0.222	0.263/0.298	0.255/0.372	0.246/0.450	0.360
General	Relaxed	0	0	0.430/0.185	0.360/0.270	0.353/0.391	0.343/0.500	0.338/0.610	0.466
Specific	Strict	0	0	0.030/0.020	0.015/0.020	0.020/0.040	0.015/0.040	0.012/0.040	0.030
Specific	Relaxed	0	0	0.080/0.040	0.065/0.060	0.063/0.080	0.052/0.083	0.050/0.100	0.083

Table 2. Experimental Results of adding Noise but no user Feedback

Randomness	Temperature	P@1/R@1	P@2/R@2	P@3/R@3	P@4/R@4	P@5/R@5	mAP
0.361	0	0.250/0.107	0.280/0.226	0.273/0.329	0.265/0.409	0.266/0.540	0.358
0.228	0	0.260/0.080	0.275/0.196	0.287/0.328	0.292/0.442	0.298/0.570	0.359
0.14	0	0.270/0.086	0.255/0.172	0.253/0.269	0.253/0.359	0.262/0.480	0.320
0.09	0	0.290/0.100	0.300/0.229	0.300/0.343	0.297/0.463	0.296/0.570	0.379
0.054	0	0.300/0.139	0.280/0.236	0.300/0.381	0.285/0.461	0.286/0.570	0.393

Table 3. Experimental Results of adding User Feedback for the **top 20 topics, with a user count of 1** and no Noise

Randomness	Temperature	P@1/R@1	P@2/R@2	P@3/R@3	P@4/R@4	P@5/R@5	mAP
0	0.667	0.440/0.202	0.370/0.301	0.350/0.412	0.340/0.517	0.328/0.600	0.474
0	1.0	0.430/0.190	0.365/0.286	0.360/0.424	0.345/0.524	0.336/0.630	0.476
0	1.429	0.440/0.203	0.365/0.296	0.350/0.410	0.335/0.507	0.328/0.610	0.475
0	1.765	0.440/0.197	0.365/0.288	0.357/0.414	0.343/0.507	0.334/0.600	0.473
0	1.961	0.410/0.178	0.350/0.267	0.337/0.378	0.328/0.481	0.318/0.570	0.444

Table 4. Experimental Results of adding User Feedback for the **top 20 topics, with a user count of 25** and no Noise

Randomness	Temperature	P@1/R@1	P@2/R@2	P@3/R@3	P@4/R@4	P@5/R@5	mAP
0	0.667	0.220/0.087	0.230/0.164	0.230/0.241	0.245/0.355	0.250/0.470	0.307
0	1.0	0.220/0.086	0.240/0.172	0.247/0.274	0.255/0.386	0.256/0.490	0.318
0	1.429	0.220/0.075	0.230/0.146	0.237/0.235	0.250/0.347	0.252/0.450	0.298
0	1.765	0.210/0.067	0.230/0.143	0.240/0.242	0.253/0.355	0.254/0.460	0.297
0	1.961	0.210/0.078	0.230/0.159	0.233/0.241	0.250/0.365	0.250/0.460	0.301

Table 5. Experimental Results of adding User Feedback for the **top 50 topics, with a user count of 1** and no Noise

Randomness	Temperature	P@1/R@1	P@2/R@2	P@3/R@3	P@4/R@4	P@5/R@5	mAP
0	0.667	0.430/0.194	0.370/0.299	0.350/0.401	0.343/0.515	0.332/0.610	0.473
0	1.0	0.420/0.191	0.365/0.300	0.347/0.403	0.340/0.513	0.330/0.610	0.470
0	1.429	0.430/0.192	0.355/0.276	0.347/0.402	0.338/0.512	0.326/0.600	0.463
0	1.765	0.430/0.191	0.365/0.290	0.350/0.403	0.345/0.523	0.332/0.610	0.470
0	1.961	0.450/0.212	0.370/0.306	0.357/0.430	0.340/0.527	0.330/0.620	0.485

Table 6. Experimental Results of adding User Feedback for the **top 50 topics, with a user count of 1 and Noise**

Randomness	Temperature	P@1/R@1	P@2/R@2	P@3/R@3	P@4/R@4	P@5/R@5	mAP
0.361	0.667	0.260/0.092	0.275/0.219	0.283/0.321	0.280/0.406	0.280/0.490	0.348
0.361	1.961	0.260/0.094	0.270/0.204	0.260/0.285	0.273/0.413	0.276/0.510	0.342
0.054	0.667	0.280/0.114	0.305/0.244	0.307/0.362	0.312/0.485	0.294/0.550	0.388
0.054	1.961	0.280/0.110	0.290/0.217	0.297/0.343	0.287/0.427	0.284/0.510	0.367

Table 7. Experimental Results of Randomly guessing matching articles

Model	P@1/R@1	P@2/R@2	P@3/R@3	P@4/R@4	P@5/R@5	mAP
Random Guessing	0.010/0.010	0.020/0.040	0.020/0.060	0.018/0.070	0.020/0.100	0.040