CSE 493S/599S Project Final Version

Justin Chae, Thomas Lilly, Johan Lindqvist, Yehchan Yoo
University of Washington
{jchae3, tlilly, lindq2, yehchany}@uw.edu

Machine Learning Project Summary

We investigate the claimed effectiveness of Rec-R1, a newly proposed reinforcement learning framework for combining large language models and recommendation systems.

Project Scope

The goal of our project was to replicate the results of the Rec-R1 paper using its existing GitHub repository. Specifically, we wanted to confirm their claims that their reinforcement learning method improved upon supervised-fine-tuning and baseline methods, was more computationally efficient than other methods, and maintained or improved the large language model's general-purpose abilities after training.

Methodology

We used the authors' provided code and datasets in the repository [1] associated with the paper to verify their findings. While this did not require creating new models or training pipelines, there was a significant amount of trial and error required to successfully train and evaluate the models. We trained sparse and dense retriever variants of the base model used for the Rec-R1 framework, Qwen-2.5-3B-Instruct. The datasets used for training included ESCI [2], Amazon C4 [3], and Amazon Beauty [3]. Since the paper's specified GPUs for training, two 80GB A100s, are not available in Hyak, we used two 48GB L40/L40s GPUs with reduced tensor parallelization. The training time required for each model varied from 7 hours to 20 hours for a single epoch. Due to time constraints, some model comparisons were made with checkpoints from a single epoch rather than the original amount in the authors' code. We did not attempt SFT result replication, because — as the authors state — the SFT results converge to the teacher, making teacher model baselines a sufficient comparison.

Results

We found that we could replicate the evaluation results of the paper, but we struggled to confirm their claim of computational efficiency offered by the framework. We were able to achieve similar evaluation scores in all of the experimental setups tried by the authors. However, we could only train our models for at most half of the epochs the authors trained for or even only for 1 epoch. Finally, we confirmed the authors' claims that general reasoning capabilities did not deteriorate after post-training and got very similar benchmark results to the authors.

What was Easy

It was easy to access the necessary materials to reproduce the study, as everything was open-source. The code-base, the datasets and the models used in this study could all be found and used through the repository created by the original authors.

What was Difficult

Replicating the Rec-R1 paper proved highly difficult in that we struggled to navigate through the poor documentation and the disorganized structure of the repository. Additionally, the paper required a lot of computational power, requiring us to use the UW Hyak clusters that were often in shortage and underpowered compared to clusters used in the paper.

1 Introduction

The goal of this project is to replicate the results of the Rec-R1 paper [4]. This paper proposed a new framework that combines large language models (LLMs) and recommendation systems (RecSys) through a closed feedback loop where a reward signal from the recommendation system updates model weights of the LLM.

This framework was proposed in light of issues from utilizing LLMs in recommendation tasks. Existing approaches such as prompting and supervised fine-tuning (SFT) see LLMs and recommendation model as separate components without connecting them with a closed feedback loop. As a result, LLMs are optimized for recommendation through proxy objectives that may not completely align with the task. For instance, one can generate SFT data using another closed source LLM or they can get higher quality data through human annotation. Both options are costly in time and money and don't have a way of obtaining a gradient update based on a feedback signal from the RecSys, leaving the LLM and the RecSys as disjoint systems.

Rec-R1 addresses these issues by joining the RecSys and LLM using reinforcement learning (RL). Given a user input, the LLM generates an input for the RecSys. The RecSys then uses a rule-based metric (e.g., NDCG, Recall) to evaluate the input which is turned into a reward signal for the LLM [4, Section 1].

The results from the paper evaluated Rec-R1 against SFT and prompting in product search and sequential recommendation tasks. The paper found that Rec-R1 significantly improves over SFT and prompting in both tasks, and Rec-R1 showed strong performance even in the cold-start scenario for sequential recommendation [4, 3.2.2 Results]. Additionally, Rec-R1 required far less time and money to achieve the same performance as SFT on these tasks [4, Appendix D]. Finally, the Qwen-2.5-3B-Instruct LLM was also evaluated to see if its general purpose reasoning capabilities were maintained after SFT and Rec-R1's RL training. The paper found that LLM reasoning dropped in quite a few situations after SFT (especially on GSM8K and IFEval) but Rec-R1 almost always maintained or improved LLM performance after RL training compared to the baseline Qwen-2.5-3B-Instruct LLM [4, Appendix E.3].

2 Scope of the Project

Our goal was to attempt to replicate and evaluate the authors' key claims that Rec-R1 can effectively optimize the development of large language model by making data distillation much more resource-efficient while improving the performance of the resulting model [4, Section 1]. Due to time and resource constraints we limit ourselves to a reproduction of the metrics for the Rec-R1 framework itself and use the reference scores from the original paper for the baselines from other frameworks and models.

2.1 Addressed Claims/Hypothesis from the Original Paper

Being more specific, we were testing the following two claims from [4, Section 1]:

- **Claim 1:** Rec-R1 achieves comparable or superior performance than prompting and SFT tools in optimizing LLM generation with less computational cost.
- **Claim 2:** Rec-R1-optimized language models improve the ability of an LLM to retain its general-purpose abilities over time.

Being even more specific, we were testing the two aforementioned claims by trying to replicate the following results from [4, Section 3]:

- **Claim 1 Result:** Rec-R1-optimized large language models (Qwen and gpt-4o, in our case) exhibit higher retrieval performance than prompting-only baselines and SFT versions of the model as measured by NDCG@100 and Recall@k for ESCI, Amazon-C4, and Amazon Beauty datasets (which will be discussed further in Section 3) [4, Section 3].
- **Claim 2 Result:** Rec-R1-optimized large language model (Qwen, in our case) achieves an NDCG@100 score as much or higher than that of the non-optimized version of the model on general-purpose reasoning datasets IFEval, GSM8K, MBPP, and HumanEval [4, Appendix E.3.2].

3 Methodology

3.1 Dataset

To replicate the Rec-R1 paper, we used the same datasets the researchers used for training and testing the model: ESCI [2] and Amazon-C4 [3] for product search, and the Amazon Beauty Dataset for the sequential recommendation [4, Sec 3.1.1 and 3.2.1]. Both Amazon Beauty and Amazon-C4 are constructed from a larger dataset, Amazon Reviews [3]. Within the ESCI [2] dataset, the authors focus on the subcategories of *Video Games*, *Baby Products*, *Office Products*, and *Sports and Outdoors*. The subcategory specific data splits contain 4510 training samples, 898 validation samples, and 798 test samples per category. The Amazon C4 [3] dataset is included because it shares the same task domain as ESCI, but the queries are expressed in natural language. This dataset only has category labels for test queries so consistent with the authors of the framework, we use the prior listed subdomains as the test set to obtain results comparable with those from the ESCI trained models. Each of the queries were modified with a template similar to Figure 1. The datasets were available through the Rec-R1 GitHub repository [1]. Although the repository included predefined splits, we created our own train, validation, and test sets to maintain control over the evaluation process and ensure consistency in our replication.

Prompt Template for REC-R1 + BM25 (Product Search)

```
You are an expert in query generation. Given a query, your task is to create
query terms to retrieve the most relevant products, ensuring they best meet
customer needs.

Below is the query:
``` {user_query} ```

<|im_start|>system
You are a helpful AI assistant. You first think about the reasoning process
in the mind and then provide the user with the answer.
<|im_end|>
<|im_start|>user
[PROMPT as above]
Show your work in <think>\think> tags. Your final response must be in JSON
format within <answer>\answer> tags. The generated query should use Boolean
operators (AND, OR) to structure your query logically. For example,
<answer>
{ "query": xxx }
</answer>.
<|im_end|>
<|im_start|>assistant
Let me solve this step by step.
<think>
```

Figure 1: Prompt Template for Rec-R1 + BM25 for Product Search

3.2 Coding

We used the authors' code [1] to attempt to reproduce their results. As the aim is to reproduce the original results, we decided to rely on the original code for implementation. However, due to issues encountered during the initial experiments, some changes have been made to the original code — mainly to resolve compatibility issues within the frameworks used. As the authors relied on older versions of the frameworks some of which are no longer available, we've been required to make some updates to the frameworks in use.

The Rec-R1 training framework is model agnostic and adapted for sparse and dense retrievers [4, Appendix E.1.2]. The sparse retriever implementation uses Pyserini [5] with Lucene's BM25 [6] implementation.

Dense retrieval uses HNSW-based FAISS indices [7]. The embeddings used for FAISS were experimented with several different embedding models, primarily BLAIR [8] and RoBERTa [9]. BLAIR and RoBERTa were also used as discriminators for comparison studies along with gpt-4o [10] and Qwen-2.5-3B-Instruct [11].

The language model used during fine-tuning was initialized from Qwen-2.5-3B-Instruct [11]. Due to resource limitations, smaller versions of the model were also used for testing during development stages of the project.
```

All of our codes are hosted on our GitHub repository `Rec-R1_magic`. The repository is a modified version of the authors' original code repository [1] and is publicly accessible at the following link: `https://github.com/yehchanyoo/Rec-R1_magic`

3.3 Computational Feasibility

The Rec-R1 framework was designed with computational efficiency as a key priority and as such did not require excessive amounts of compute. The GPUs used for training the models in the original study were two Nvidia A100 with 80 GB of VRAM. The memory utilization of these GPUs was described as capped to 30 percent of the total capacity of these GPUs, with the runtime for the fine tuning limited to only 210 seconds using this setup [4, Appendix D]. The measurement of time for fine tuning was mentioned to be the time necessary to match the performance of a model trained for two epochs using SFT. The two epoch SFT ran for 35 minutes on the authors setup, but training details in terms of number of steps required to achieve comparable performance using Rec-R1 was not mentioned.

For our computational resources, we've used Hyak, where we have access to L40 and L40s GPUs through the RCC account [12]. These have less memory compared to the A100's used in the original study; however, due to the low utilization of the GPUs in the original paper, replicating limited results of the paper was feasible [4].

Our experiments showed that the framework ran with the Qwen-2.5-3B-Instruct [11] using the distributed mode on two L40 or L40s GPUs. A minor author-proposed modification was necessary to make the code more efficient in VRAM usage. A single GPU proved to be insufficient with out-of-memory errors, which required us to limit the model size to the Qwen-2.5-0.5B-Instruct [11] for initial testing. The OOM issues encountered were unexpected, as we initially estimated that a single GPU would be sufficient considering the descriptions of VRAM usage in the original paper. The reason behind this appears to be the number of jobs parameter, which was originally set to 12 but was reduced to 2 for our training. This also reduced the time required for training on L40/L40s GPUs, as it reduced the time required for moving data between the GPUs to the CPUs. However, the time necessary for fine-tuning still far exceeded the time noted in the original paper, which was likely a consequence of the lower VRAM.

4 Results/Summary

4.1 ESCI

We recorded the normalized discounted cumulative gain for the top 100 items retrieved (NDCG@100), which measures how well items were ranked as well as how relevant they are. We found that we could replicate the original paper's results with at worst about 8% off of the paper's results with only 5 epochs of training compared to their 10 epochs of training, as shown in Table 1.

Table 1: NDCG@100 for Sparse Retrieval on ESCI

Model	Video Games	Baby Products	Office Products	Sports & Outdoors
BM25	12.44	15.12	23.96	19.48
GPT-4o + BM25	26.06	23.05	27.98	27.38
Qwen-2.5-3B-Instruct + BM25	19.63	16.03	19.96	21.36
Rec-R1-3B + BM25	**33.89**	**29.27**	**34.61**	**31.92**
Rec-R1-3B + BM25 (Repro)	26.16	27.91	34.38	30.99

4.2 Amazon C4

The authors used the Amazon C4 [3] dataset to measure how models optimized with the proposed framework perform on a complex natural language product search task. They also used NDCG@100 as the evaluation metric for this dataset to allow easy comparisons with the ESCI results [4, Section 3.1]. Given the computation and time constraints discussed in Section 3.3, we decided to only train the model for 1 epoch and only use BLAIR-BASE dense retriever. The results we obtain in Figure 2 for the BM25 retriever show impressive gains by nearly doubling the best-performing sparse baselines. However, these results still fall between 19% and 25% below the NDCG scores the authors obtained for the test subsets. The incremental gains over one epoch suggest that the scores presented are attainable over approximately 5 epochs of training. However, we remain doubtful of the computational efficiency claims in Figure 1 (b) of the paper. The authors claim that tuning Qwen-2.5-3B-Instruct only took 210 seconds [4, Figure 1]. But our time measurements of 11 hours for the sparse retriever and 7 hours for BLAIR-BASE resemble

the SFT training times in Figure 1(b) of the original paper [4, Figure 1]. Similar to the authors, we will not compare our results directly to the SFT baselines, since the learned policies converge to the teacher model baseline [4, Theorem 1].

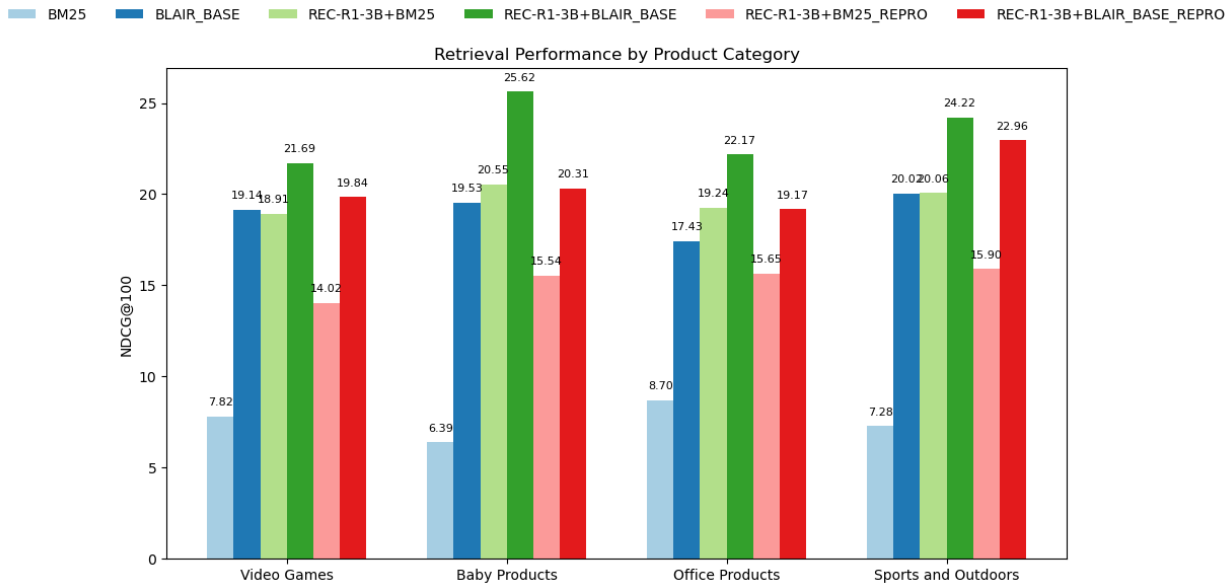

Figure 2: Amazon C4 Results

4.3 Amazon Beauty

The authors also use the Amazon Beauty dataset to test how Rec-R1 optimized models perform on sequential recommendation tasks. Sequential recommendation tasks involve finding the most relevant item to recommend next to a user based on that user's past purchase history. Here, the authors also use NDCG and Recall metrics to evaluate the recommendation results [4, Section 3.2].

Due to a lack of computational resources, we decided to train the model for 1 epoch only and for a maximum of 400 steps using the BM25 retriever and the frozen Qwen-2.5-3B-Instruct LLM.

We also ran the training process under both **transductive** and **inductive** settings.

- **Transductive** setting: Test items are in the training set [4, Section 3.2.1].
- **Inductive** setting: Test items are *not* in the training set [4, Section 3.2.1].

Despite the limited number of runs and epochs, we were able to achieve scores that were fairly similar to the scores achieved in the paper with 50 full epochs [1]. The results can be seen in Figure 3. While the replication results had lower scores in both settings for all metrics (Recall@, NDCG@), the replication results had metrics fairly close to the actual results — especially for NDCG@ scores under transductive settings. Additionally, similar to the results stated in the original paper, the replication results showed the model struggling with sequential recommendation (with less than 10% scores), especially under transductive settings with lower scores across the board than under inductive settings [4, Section 3.2.2].

4.4 Generalization

Despite the lack of evidence for the authors' reported computational efficiency, we did find evidence for their claims that models optimized using the Rec-R1 framework retained general-purpose abilities better than SFT optimized models [4, Section 3.3]. In Table 2, we saw that the Rec-R1 model trained using the BLAIR-BASE dense retriever was able to score higher than the Qwen Instruct baseline and match the authors' Rec-R1 IFEval score. Other scores were similarly close to the authors or higher. Considering these models were trained for much less epochs than the authors' models, we expected the scores to fall somewhere between the baseline and the authors' model.

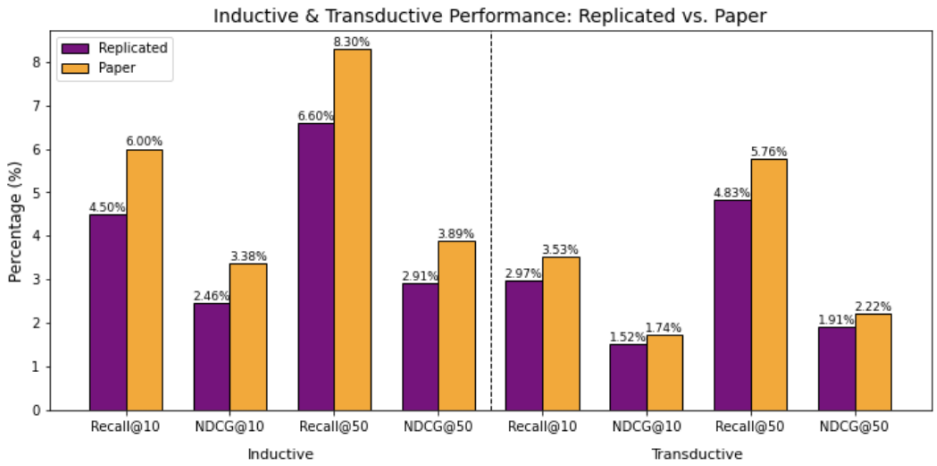

Figure 3: Amazon Beauty Results: Replication vs. Paper

Table 2: Model Generalization Across Benchmarks

Model	ESCI	MMLU	IFEval	GSM8K	MBPP	HumanEval
Qwen-2.5-3B-Instruct (Baseline)	19.3	**65.4**	58.2	63.4	53.6	46.3
Qwen-2.5-3B-Instruct (SFT)	24.4	63.7	31.4	35.7	54.8	**53.6**
Rec-R1-3B-ESCI+BM25 (Original)	**32.2**	65.3	60.1	**69.1**	54.4	46.3
Rec-R1-3B-C4+BM25 (Repro)	–	65	58	64	–	–
Rec-R1-3B-ESCI+BM25 (Repro)	–	**65.4**	**61.4**	67.3	**55.2**	51.2
Rec-R1-3B+BLAIR-BASE (Repro)	–	–	60	66	–	–

5 Discussion

Overall, we were largely able to replicate the results presented in the original paper with some shortcomings.

- **Claim 1:** We achieved similar results to the authors on NDCG@k and Recall@k and confirm that Rec-R1 achieves comparable or superior performance to prompting and SFT. However, we did not achieve the low computational cost for the framework from the authors and were often limited by our computational resources.
- **Claim 2:** We confirmed that the Rec-R1 framework leads to maintained or improved general-purpose capabilities after post-training.

5.1 What was Easy

The replication of the study was made easier by the fact that all of the material required to reproduce the study was publicly available. The code required for all parts of the replication is available at a public GitHub repository [1]. The structure of the repository and the code was far from ideal, but the fact that it was available made it possible for us to mainly rely on their code rather than writing our own. The scripts written by the original authors enabled us to use their scripts for data processing, model training and evaluations on all tasks in this paper. For a large number of them, slight tweaks were necessary to handle inconsistencies, but otherwise we did not need to rewrite the underlying training and retrieval functions. The datasets were also available through online resources, which meant there was no need for the otherwise potentially expensive replication process of regenerating the query files. Details for model training was included in the extensive appendix, which helped us verify the scripts for training and evaluation.

5.2 What was Difficult

Replication of the Rec-R1 paper presented numerous challenges — so much so that we were not able to replicate the paper to its full extent.

First, the poor documentation and structure of the code base inside the repository significantly impeded our ability to replicate the paper. With the exception of the Amazon-C4 product search task, the repository did not document any explicit instruction for replicating the paper's tasks. Even for the Amazon-C4 product search task, the authors' instruction on `README.md` did not include information on what to do after training. Additionally, the packages suggested in the authors' code often had an outdated or incompatible version for our code, especially for Python and JDK.

Also, the repository did not delete outdated scripts and did not provide any information on which scripts should be used for replication. For instance, for performing data preprocessing for the Amazon Beauty sequential search, the repository had two almost identical files in `src/dataset/amazon_review/inst/sparse` that performed the same task. We only found through looking at the codes and testing out the scripts that only one of them was usable for this task and the other one was too outdated. The author marked the outdated file's name with an underscore suffix at the end, but did not indicate anywhere else on the repository that this file was outdated. The author even unnecessarily complicated the repository's structure by embedding the entire Amazon Reviews 2023 repository [13] within the Rec-R1 repository rather than providing a script for users to download this repository on their own — making the Rec-R1 repository unnecessarily difficult to navigate around. Such poor documentation and poor repository structure led to much of the project time being spent on debugging and figuring out the file structure, significantly hindering our progress in replicating the paper.

Additionally, it was often difficult to obtain the needed GPU and CPU power for replicating this paper. The computational demands of Rec-R1 made replication infeasible on personal devices, so we relied on UW's Hyak cluster for access to powerful GPUs. However, even for the most powerful tasks, we found that UW Hyak only had a maximum of 8 L40's and 8 L40S's available at any time in the free `stf` partition for the entire University of Washington student body, while other GPUs in Hyak were either unavailable to us or too weak to use for replication. While the TAs have recommended utilizing 4 L40's or 4 L40S's to replicate the paper, it was practically impossible to get that many GPUs in a Hyak session due to a consistent lack of L40's and L40S's available for the entire student body in the Hyak clusters, and we had to perform training with only 2 L40's or 2 L40S's at best. This lack of GPU compute made it difficult to perform long training tasks in the paper, and we had to cut down the number of training epochs and steps to get practical replication results. Additionally, UW Hyak's GPU clusters utilized Apptainer instead of Docker for containerization, and we had to optimize the existing code to not only work with the lower graphical power but also with the use of Apptainer in place of Docker — further hindering our progress in replicating the paper.

5.3 Recommendations for Future Work

For future work, training using the same hardware resources would be a priority. We did not see the performance in terms of speed that the authors reported for Rec-R1, and to what degree that was caused by the lack of VRAM in the GPUs used for our reproduction is still unknown. To successfully reproduce the efficiency component of Rec-R1, training on A100 GPUs would be critical.

Further testing should also require training for longer to match the authors' numbers of epochs. Our reported scores were generally worse than their results, which was likely caused by fewer number of epochs and training steps.

However, the main issue with the attempts to reproduce this study is that Rec-R1 might simply not be mature for a replication at this stage. The study is in preprint and it appears as if the authors of the paper are performing further experiments. The code is still seeing occasional updates, which is likely contributing to the lack of organization in the repository. Further studies working on it should wait until the paper itself is finished and the repository is left static by which the documentation will also hopefully be improved. A significant cleanup of the code base would go a long way in streamlining future replications.

The results we see still show that Rec-R1 does present a promising direction for future research; and, when the study is entirely finished, a full replication should be conducted to verify the complete results. In a larger scope, we believe that our study confirms that reinforcement learning with GRPO is a promising method for fine-tuning models on specialized tasks while maintaining good generalized performance.

References

[1] J. Jin, *Rec-r1*, https://github.com/linjc16/Rec-R1/tree/main, Accessed: 2025-04-24, 2025.

[2] C. K. Reddy, L. Màrquez, F. Valero, *et al.*, "Shopping queries dataset: A large-scale ESCI benchmark for improving product search," 2022. arXiv: 2206.06588.

[3] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, and J. McAuley, "Bridging language and items for retrieval and recommendation," *arXiv preprint arXiv:2403.03952*, 2024.

[4] J. Lin, T. Wang, and K. Qian, *Rec-R1: Bridging Generative Large Language Models and User-Centric Recommendation Systems via Reinforcement Learning*, arXiv:2503.24289 [cs], Mar. 2025. DOI: 10.48550/arXiv.2503.24289. [Online]. Available: http://arxiv.org/abs/2503.24289 (visited on 05/18/2025).

[5] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira, "Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations," in *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, 2021, pp. 2356–2362.

[6] J. Pérez-Iglesias, J. R. Pérez-Agüera, V. Fresno, and Y. Z. Feinstein, *Integrating the probabilistic models bm25/bm25f into lucene*, 2009. arXiv: 0911.5046 [cs.IR]. [Online]. Available: https://arxiv.org/abs/0911.5046.

[7] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021. DOI: 10.1109/TBDATA.2019.2921572.

[8] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, and J. McAuley, *Bridging language and items for retrieval and recommendation*, 2024. arXiv: 2403.03952 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2403.03952.

[9] Y. Liu, M. Ott, N. Goyal, *et al.*, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1907.11692.

[10] OpenAI, : A. Hurst, *et al.*, *Gpt-4o system card*, 2024. arXiv: 2410.21276 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2410.21276.

[11] Q. Team, *Qwen2.5: A party of foundation models*, Sep. 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/.

[12] U. of Washington Research Computing Club, *Start here*, https://hyak.uw.edu/docs/gpus/gpu_start/, Accessed: 2025-04-24.

[13] Y. Hou, *Amazon reviews 2023*, https://github.com/hyp1231/AmazonReviews2023, Accessed: 2025-06-06, 2023.