Beyond Parameter Counts: Benchmarking Similar-Sized Large Language Models for Next-Item Recommendation

Kavach Dheer, Peter Corcoran and Josephine Griffith
University of Galway, Ireland
fi

Keywords: Large Language Models, Recommender Systems, Next-Item Prediction, Model Benchmarking, Data Leakage Analysis.

Abstract: Large language models (LLMs) are rapidly being integrated into recommender systems. New LLMs are released frequently, offering numerous architectures that share identical parameter sizes within their class, giving practitioners many options to choose from. While existing benchmarks evaluate LLM-powered recommender systems on various tasks, none have examined how same-sized LLMs perform under identical experimental conditions as a recommender system. Additionally, these benchmarks do not verify whether the evaluation datasets were part of the LLMs pre-training data. This research evaluates five open-source 7–8B parameter models (Gemma, Deepseek, Qwen, Llama-3.1, and Mistral) using a fixed A-LLMRec architecture for next-item prediction using the Amazon Luxury-Beauty Dataset. We measure top-1 accuracy (Hit@1) and evaluate dataset leakage through reference-model membership-inference attacks to ensure no model gains advantages from pre-training exposure. Although all models show negligible dataset leakage rates ($< 0.2\%$), Hit@1 varies dramatically across 20 percentage points, from 44% for Gemma to 64% for Mistral, despite identical parameter counts and evaluation conditions. These findings demonstrate that selecting among the most appropriate LLMs is a crucial design decision in LLM-based recommender systems.

1 INTRODUCTION

Recommender systems now power a wide range of applications, from e-commerce search engines to large-scale video-streaming platforms. State-of-the-art LLMs have transformed artificial intelligence capabilities showing remarkable performance in natural language processing, robotics, information retrieval, and multi-modal computer vision tasks. Motivated by these successes, a growing body of work has begun to integrate LLMs with recommender system architectures. Empirical studies (Petrov and Macdonald, 2023; Bao et al., 2023; Geng et al., 2022) demonstrate that LLM-augmented recommender systems can substantially outperform conventional baselines, sparking considerable interest in this hybrid research direction.

LLMs are being released at an unprecedented pace, with model sizes ranging from a few hundred million to over one hundred billion parameters. Many of these models are open-source and publicly available via platforms such as Hugging Face, while major AI companies roll out closed-source LLMs through their own APIs. The wide availability of LLMs offers practitioners a rich set of choices while introducing new challenges in model selection.

The research into integrating LLMs into recommender systems has largely focused on surrounding innovations, for example, improving architectural frameworks (Geng et al., 2022), refining data pipelines (Borisov et al., 2022), or developing advanced prompt engineering strategies (Diao et al., 2023) and fine-tuning approaches (Bao et al., 2023). Far less attention has been paid to the fundamental question of how the choice of the LLM itself influences recommendation quality. In other words, prior work rarely examines LLM selection as an isolated variable, leaving a gap in understanding how different LLMs of the same parameter size might affect recommendation results and quality. Existing recommender system benchmarks (Wu et al., 2024; Geng et al., 2022) compare unequally sized models and do not test for data leakage, leaving the question of how model choice alone impacts recommendation accuracy unanswered. With numerous LLMs now available at similar parameter size, it is increasingly difficult for practitioners to determine which model to adopt for a given task. It is often assumed that all

Dheer, K., Corcoran, P. and Griffith, J.
Beyond Parameter Counts: Benchmarking Similar-Sized Large Language Models for Next-Item Recommendation.
DOI: 10.5220/0013736800004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 1: KDIR*, pages 364-371

models of the same scale will perform comparably, yet this assumption lacks empirical validation and may not hold true in practice. In fact, a suboptimal LLM choice can degrade a recommender system's accuracy, whereas selecting the right model, even within the same parameter size, can significantly improve overall performance. This underscores an urgent need for a systematic examination of model choice as a factor in recommender system accuracy. A second, frequently overlooked issue is data leakage. If the evaluation corpus forms part of an LLM's pre-training data, the model may appear to excel simply because it has already been trained on the dataset. Without validating if this potential leakage has occurred, studies may mistake memorization for genuine modeling capability. Therefore, to fairly evaluate model performance, we must first determine whether each candidate LLM's training data includes the evaluation dataset, i.e., whether data leakage is present.

Accordingly, we pose two research questions:

- **RQ1:** Does the choice of 7-8B parameter LLMs significantly affect recommendation performance ?
- **RQ2:** How can we assess and quantitatively measure the extent to which overlap between an LLM's pre-training data and the evaluation corpus inflates its observed recommendation accuracy?

Our main contributions are summarized as follows:

- **Model-Isolation:** Holding the pipeline fixed, we benchmark five open-source 7–8B parameter LLMs (Mistral, Gemma, Deepseek, Qwen, and Llama-3.1) on the Amazon Luxury Beauty dataset (Ni et al., 2019), reporting Hit@1 for the next-item-prediction task.
- **Data-Leakage Analysis:** For every data instance we run a reference-based membership-inference attack to estimate whether the data instance appeared in the model's pre-training corpus, allowing us to quantify and contextualize the impact of memorization on the observed model accuracy.

By holding architecture conditions constant while verifying pre-training leakage, we find that switching between 7–8B parameter LLM models can change top-1 recommendation accuracy by 20 percentage points, from 44% for Gemma to 64% for Mistral. These differences persist even after accounting for the memorized items (data leakage). This demonstrates that model choice alone affects recommendation accuracy.

The remainder of this paper is organized as follows: Section-2 reviews related work; Section-3 details our methodology; Section-4 presents results and discussions; Section-5 concludes.

2 RELATED WORK

The progression of LLM-based recommender systems can be divided into three distinct stages.

1. **Utilizing LLMs for Zero-Shot and Few-Shot Recommendations:** Several studies have explored whether off-the-shelf LLMs can rank candidates or make next-item suggestions without task-specific training. Hou et al. (2024) demonstrates that LLMs can function as zero-shot rankers with careful prompting, while Wang and Lim (2023) develops a prompting strategy for zero-shot next-item recommendation. LLMs are also valuable for providing explanations for their recommendations and serving as conversational recommender systems. For instance, Gao et al. (2023) employs ChatGPT to offer reasoning behind its recommendations.
2. **Fine-Tuning LLMs for Recommendations:** To enhance performance, researchers began fine-tuning LLMs for specific tasks. Friedman et al. (2023) fine-tuned Llama for YouTube recommendations, while Liao et al. (2024) and Chen et al. (2024) adapted LLMs for sequential recommendation tasks to better interpret user behavior patterns. Meanwhile, Geng et al. (2022), proposed a model(P5) that casts recommendation as a text-to-text problem and unifying tasks (e.g., next-item prediction, rating, explanation, review summarization) in a single T5-based model via personalized prompts.
3. **Benchmarking LLMs for Recommendation Tasks:** As efficient fine-tuning techniques like LoRA emerged and new LLMs proliferated, research moved towards benchmarking these various models across recommendation tasks. Liu et al. (2025) creates a benchmark that evaluates conventional recommender systems against LLM-integrated recommender systems and assesses two tasks, click-through rate prediction (CTR) and sequential recommendation, evaluating 17 LLMs (of different sizes) across 5 datasets. Additionally, Liu et al. (2023) evaluates LLMs (of different sizes) on 5 different recommendation tasks: rating prediction, sequential recommendation, direct recommendation, explanation generation, and review summarization, focusing on LLM performance across different tasks. Xu et al. (2024) builds upon Geng et al. (2022)'s method to evaluate sequential recommendation, though they only utilize two LLMs: T5 and Llama-2. Additionally, Wu et al. (2024) focuses on cold-start scenarios, evaluating and comparing LLMs and traditional

deep learning recommender systems solely using semantic text for zero-shot recommendation. However, the LLMs they utilize range from 355M to 7B parameters, without isolating and comparing LLMs of the same size. Liu et al. (2024) focused on optimizing a single LLM for conversational recommendation scenarios, though this narrow focus limited broader comparative insights, whereas Jiang et al. (2025) uses various evaluation dimensions (including history length, candidate position bias, user profile generation, and hallucinations) while also fine-tuning the LLMs.

Despite impressive progress, no prior study isolates the effect of the LLMs themselves when all other variables are held constant. We compare five open 7-8B parameter LLMs under identical conditions. This controlled approach reveals true performance differences between LLMs.

3 METHODOLOGY

3.1 Baseline Architecture

We build on A-LLMRec (Kim et al., 2024) which is an LLM-based recommender system that aligns a frozen pre-trained collaborative filtering recommender system with an LLM. It adds a lightweight alignment network which is the only part that is trained: this network transforms the user and item embeddings produced by the collaborative filtering backbone into the token space of the LLM, so neither the collaborative filtering model nor the LLM itself needs fine-tuning.

We chose A-LLMRec because it excels in both cold and warm scenarios. Since we want to test LLM performance, we select this architecture to assess the capabilities of conventional recommender systems and LLMs together, ensuring both components enhance rather than degrade performance due to their respective limitations. Additionally, A-LLMRec freezes both the collaborative filtering recommender system and the language model, learning only a lightweight alignment network; this design isolates the LLM so that any language model can be swapped in without re-training the whole recommender itself. Meanwhile, A-LLMRec reports faster training/inference and state-of-the-art accuracy across several benchmarks. These traits make A-LLMRec an efficient, reproducible starting point for investigating LLM variations.

Our implementation uses Self-Attention Sequential Recommendation (SASRec) (Kang and McAuley, 2018) as the collaborative filtering backbone. We replace the original LLM (OPT-6.7B) with five alternative LLMs, all other components follow the original setup.

A-LLMRec involves two pre-training stages:

1. **Stage-1 Alignment between Collaborative and Textual Knowledge:** As shown in Figure 1, Stage-1, aligns the item embeddings from a frozen SASRec with their associated text information to capture both collaborative and textual knowledge to obtain the joint collaborative-text embedding. To extract text embeddings from textual information associated with items, a pre-trained SBERT (Reimers and Gurevych, 2019) is utilised. Two single-layer MLP encoders, one for items and one for text, create alignment between the item embeddings (from the frozen SASRec) and the text embeddings (from SBERT). Then the model performs latent space matching between item embeddings and text embeddings. Finally, two decoders are added to prevent the encoders from producing over-smoothed representations (where encoders produce similar outputs).
2. **Stage-2 Alignment between Joint Collaborative-Text Embedding and LLM:** As shown in Figure 2, Stage-2, aligns the joint collaborative-text embeddings obtained from Stage-1 with the LLM's token space by projecting 2-layer MLPs to the user representation and the joint collaborative-text embeddings to the token space of the LLM. This alignment enables the LLM to make recommendations by utilizing the collaborative knowledge through prompts.

3.2 LLM Models and Prompt Design

Within the A-LLMRec pipeline, we evaluate five 7-8B parameter LLMs, which are decoder-only transformers. This choice:

i) Controls for parameter scale.

ii) Matches recent high performing releases.

iii) Incorporates diverse companies to avoid bias toward a single training approach.

The architectural specifications for each model are detailed in Table 1, with all models used in frozen, inference-only mode.

We adopt the original A-LLMRec prompt design without modification. A projected user-embedding token is placed at the start of the input, followed by a natural-language instruction that lists the user's past interactions and the candidate items, each augmented with its projected joint embedding. The identical template is supplied to every LLM evaluated in

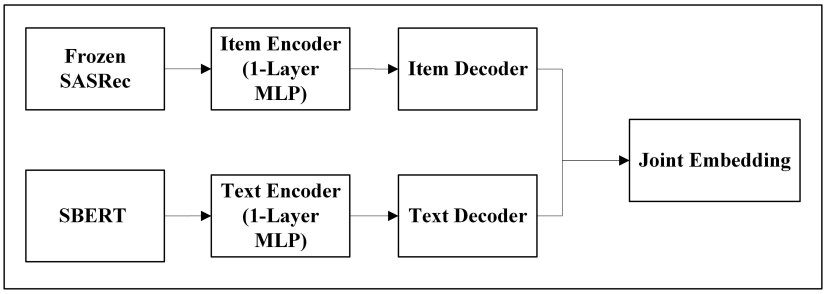

Figure 1: Stage-1: Alignment between Collaborative and Textual Knowledge.

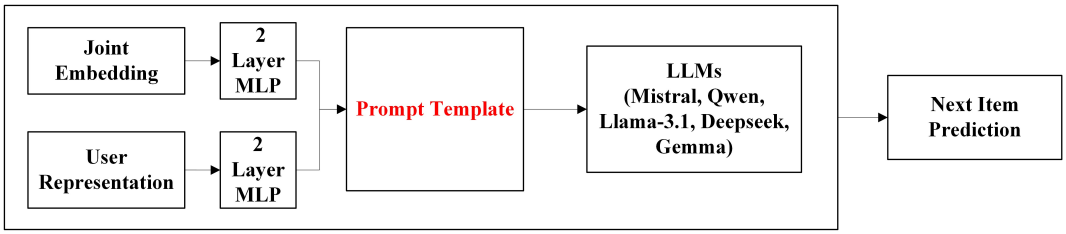

Figure 2: Stage-2: Alignment between Joint Collaborative-Text Embedding and LLM.

this study; consequently, any performance difference arises solely from the choice of model.

3.3 Dataset

We conduct all experiments on the Amazon Luxury Beauty dataset (Ni et al., 2019), which comprises a large corpus of product data including reviews, timestamps, descriptions, images, price metadata, and other attributes. The dataset originates from Amazon and covers activity from May 1996 through October 2018. We focus on Luxury Beauty, a top-level category notable for its high sparsity and variability. Following existing studies (He and McAuley, 2016; He et al., 2017), we select 9930 users and 6141 items, discarding those with fewer than 4 interactions which results in 63,953 interactions. For our setup, we utilize the title and description fields from the dataset.

3.4 Evaluation Pipeline

3.4.1 Data Split

We evaluate the models on a next-item prediction task using a temporal split by user history. For each user, we partition their interaction sequence into training, validation, and test portions in chronological order. The last item the user interacted with is held out as the test item, the second-last item is used for validation (e.g., for early stopping), and all earlier interactions form the training sequence for that user. This ensures that each test instance asks the model to recommend an item that truly comes after the training history, avoiding any future data leakage. We apply this splitting strategy to every user, so the data sets are disjoint in terms of interaction events .

3.4.2 Ranking and Metrics

We formulate the recommendation evaluation as a ranking problem. For each test case (a user and their held-out next item), we construct a candidate set consisting of the one true positive item and 19 negative items. Negatives are randomly sampled items that the user has not interacted with, drawn from the same item corpus. The task for the model is to rank these 20 candidates and retrieve the correct item at the top. We feed the user's history and the list of 20 candidate item titles (with embeddings) into the prompt as described, and allow the model to generate its single-item recommendation. The output is then matched against the candidate list. The metric we used is Hit@1 which is scored if the model's top (and only) recommendation matches the ground-truth item exactly, essentially measuring accuracy in picking the correct item out of 20.

3.5 Data Leakage & Analysis

3.5.1 Problem Definition and Review Corpus

To determine whether the Amazon Luxury Beauty dataset was included in the target model's pre-training corpus, we employ a membership-inference approach (Xie et al., 2024) on the individual product reviews from the dataset. We compare how confidently

Table 1: Core architectural specifications of the evaluated models. Params: total parameters (B=billions); L: Transformer layers; d: hidden dimension; H: attention heads; Ctx: maximum context window in thousands (k) of tokens; Tokens: approximate pre-training corpus size (T=trillions). — denotes not publicly disclosed.

Model version name	Params (B)	L	d	H	Ctx (k)	Tokens (T)
Mistral-7B-v0.3	7.25	32	4096	32	8	—
Qwen2.5-7B	7.62	28	3584	28	128	18
deepseek-llm-7b-base	7.00	30	4096	32	4	2
gemma-7b	8.54	28	3072	16	8	6
Llama-3.1-8B	8.03	32	4096	32	128	15

the target language model and a reference language model evaluate the log-probabilities of the same texts. When a model has seen a review during training, it typically assigns that review text a much higher probability (i.e., it is less surprising) than a model that has never encountered it. This two-model likelihood comparison is consistent with established membership-inference techniques, which use model loss (negative log-likelihood) as a signal of training-set membership (Xie et al., 2024). To refine this comparison, we next compute a Normalised Log-Likelihood Metric, making the two model scores invariant to sequence length and redundancy. We then set a decision threshold on the resulting likelihood gap via five-fold cross-validation, which bounds the nominal false-positive rate when flagging potential leakage.

3.5.2 Reference Model Selection

The reference model is a critical component of our method. This model must not include the dataset in its pre-training corpus since it serves as a baseline for comparison with the target model. We selected EleutherAI's Pythia-6.9B-deduped (Biderman et al., 2023) as our reference model. This 6.9B parameter language model was trained solely on the Pile dataset (Gao et al., 2020) an 825 GiB curated text corpus containing 22 diverse sources. Crucially, the Pile dataset excludes the Amazon Dataset, ensuring Pythia-6.9B has never been exposed to this data. These characteristics, combined with its parameter size matching our target LLMs, make it an ideal reference model.

3.5.3 Normalised Log-Likelihood Metric

Let $\mathbf{x} = \langle x_1, \ldots, x_T \rangle$ be a product review comprising of T tokens. We evaluate $\mathbf{x}$ using two language models:

- **Target Model:**

$$L_{\text{target}}(\mathbf{x}) = \sum_{t=1}^{T} \log p_{\text{target}}\left(x_t \mid x_{<t}\right) \qquad (1)$$

whose pre-training *may* have included the review;

- **Reference Model:** $L_{\text{ref}}(\mathbf{x})$, computed identically to reference model.

Length-Normalised Score: Raw token log-likelihoods increase (in magnitude) with sequence length and with repeated n-grams, so a straight average $L_M(\mathbf{x})/T$ is still confounded by redundancy. Following the *zlib-entropy* baseline introduced by Carlini et al. (2021), we divide by the byte length of the sequence after maximum (level 9) `zlib` compression, because lossless compression removes exactly the redundant statistics that language models try to capture. Thus, the compressed size $|\texttt{zlib}(\mathbf{x})|$ serves as a data-driven proxy for the intrinsic information content of $\mathbf{x}$. The resulting score:

$$S_M(\mathbf{x}) = \frac{L_M(\mathbf{x})}{|\texttt{zlib}(\mathbf{x})|} \quad \text{[nats per byte]} \qquad (2)$$

is length- and redundancy-invariant: higher (less-negative) values indicate that model M assigns greater probability mass to each informative byte.

Sign Interpretation:

- **Short Compressed Length:** high redundancy $\Rightarrow$ *smaller* denominator $\Rightarrow$ $S_M(\mathbf{x})$ moves *toward zero* (less-negative), i.e. better.
- **Long Compressed Length:** high entropy $\Rightarrow$ *larger* denominator $\Rightarrow$ $S_M(\mathbf{x})$ moves *away from zero* (more-negative), i.e. worse.

Thus, when the target model's per-byte log-likelihood is numerically larger (i.e., less negative) than the reference model's per-byte log-likelihood, it assigns a higher probability to the text. We capture this with:

$$\Delta(\mathbf{x}) = S_{\text{target}}(\mathbf{x}) - S_{\text{ref}}(\mathbf{x}) \qquad (3)$$

such that $\Delta(\mathbf{x}) > 0$ flags potential training-set exposure. A fixed decision threshold on $\Delta(\mathbf{x})$ converts the continuous score into the binary *seen/unseen* label used in our leak statistics. To determine the decision threshold on Δ, we apply five-fold cross-validation on the review corpus: within each fold, 80 % of reviews form a training set and the remaining 20% form the test set.

Threshold Calibration (Five-Fold Cross-Validation): For each of five random 80/20 splits, we first compute on the 80% calibration split C:

$$\mu = \underset{\mathbf{x}\in C}{\text{median}}\,\Delta(\mathbf{x})$$
$$\sigma = 1.4826\,\underset{\mathbf{x}\in C}{\text{MAD}}\big|\Delta(\mathbf{x}) - \mu\big| \tag{4}$$

where μ is the center of the calibration Δ scores and σ their robust scale (MAD = median absolute deviation). With these statistics fixed, we set a one-sided k-sigma control limit:

$$\tau = \mu + 2.33\,\sigma \tag{5}$$

choosing $k = 2.33$ to bound the nominal false-positive rate at $\approx 1\,\%$. We chose 1% to prioritize precision (avoiding false "leak" flags) given that leaks are rare and false positives are costly. The frozen threshold τ is then applied to that fold's 20% hold-out slice; leak proportions from the five folds are averaged to give an unbiased prevalence estimate, and their standard deviation provides a reliability band.

4 RESULTS & DISCUSSION

4.1 Does the Choice of Same-Sized 7–8B Parameter LLMs Affect Recommendation Performance?

We find that the choice of LLM leads to significant differences in recommendation success (Table 3 summarises the Accuracy (Hit@1) and data-leakage results). Mistral achieves the highest top-1 accuracy, with a Hit@1 of 64%. The next-best model, Llama-3.1, reaches 58%. Other models trail behind; Deepseek and Qwen offer moderate Hit@1 (around 53–52 %), and Gemma performs worst at only 44%. Notably, these performance gaps arise even though all models are of similar size (7–8B parameters), underscoring that LLM choice alone can drive divergent outcomes. Each test instance used a single fixed draw of 19 randomly sampled negatives; the reported Hit@1 values are point estimates under this protocol.

4.2 Does Pre-Training Exposure Inflate Observed Accuracy?

Data-leakage analysis reveals that none of the models significantly had seen the dataset in their pre-training corpus, the highest flag rate is merely 0.2%, with three of the five models at or below 0.06%. Since these values fall well below the 1% nominal false-positive threshold, we can confidently attribute the observed Hit@1 differences to the LLMs inherent capabilities rather than any unfair advantage from pre-training exposure to the dataset.

Table 2: Recommendation performance with dataset leakage.

Model	**Hit@1** (%)	**Dataset Leakage** (%)
Mistral	64.0	0.06
Llama-3.1	58.0	0.20
DeepSeek	53.0	0.04
Qwen	52.0	0.05
Gemma	44.0	0.00

4.3 Model Comparison

Deepseek has the lowest pre-training data (2 trillion tokens) yet performs better than Qwen, which has the largest pre-training data (18 trillion tokens). This suggests that simply adding more pre-training data does not necessarily improve accuracy.

We examined the architecture of each LLM to understand their performance differences. Unlike other models, Gemma employs multi-head self-attention with independent key/value projections and the Gaussian Error Linear Unit (GeGLU) activation function, while Mistral, Deepseek, Qwen, and Llama-3.1 use GQA and Swish Gated Linear Unit (SwiGLU) activation function.

We hypothesize that Gemma's relatively lower performance may stem from its architectural design. In grouped-query attention, the attention heads are partitioned into groups sharing key/value projections. This design trades off a small loss in representational richness for greatly reduced computational and memory cost. GQA achieves accuracy very close to full multi-head attention while substantially speeding up inference. Thus, the difference between Gemma full multi-head attention and grouped-query approach could affect model performance and efficiency. Another architectural difference lies in the feed-forward activation function. Gemma uses the GeGLU gating variant (a GELU-based GLU), whereas other models typically use SwiGLU (a Swish-based GLU). Shazeer (2020) reported that GeGLU and SwiGLU achieved nearly identical perplexities (1.942 vs. 1.944) when matched for computational cost, i.e., parameter count and FLOPs are held constant. Therefore, we would not expect a large performance gap solely from the choice of GeGLU vs. SwiGLU, though minor differences in training dynamics or convergence might exist.

Mistral combines grouped-query attention with sliding window attention, enabling better handling of

longer sequences. This enhanced capability for processing large prompts may explain why Mistral outperforms the other models.

However, we emphasize that this reasoning is speculative, we have not performed targeted ablations to isolate these effects. Our interpretation represents a plausible hypothesis grounded in known architectural trade-offs, not a confirmed explanation. Further controlled experiments are needed to validate whether these specific design choices cause the observed performance gap.

Our results demonstrate that LLM selection plays a crucial role in recommender system performance. This factor has been largely overlooked, as the prevailing assumption suggests that any LLM of similar parameter size would perform equivalently. While much attention has focused on architecture, prompting strategies, and fine-tuning approaches, the importance of LLM selection within the same parameter class has been underestimated. With numerous options now available at each parameter size, choosing the right model can significantly impact accuracy.

5 CONCLUSION

In our comparative study, we tested five 7-8B parameter LLMs on an identical task with the same architecture and dataset to examine accuracy variations, an aspect often overlooked in benchmarks and research when selecting LLMs of similar size. Our recommender system testing revealed Hit@1 accuracy variations of nearly twenty percentage points, with Mistral outperforming other models. Despite sharing the same parameter count, these models demonstrated that model selection alone can meaningfully influence accuracy. Through membership inference attacks, we verified that no model benefited from having the dataset in their pre-training corpus, confirming that the observed accuracy gaps represent genuine differences in model capability. We speculated that these variations stem from architectural differences. Future work should isolate the causes by evaluating across additional datasets, quantifying robustness via repeated negative-sampling draws with 95% confidence intervals, and examining the effects of fine-tuning and scaling to larger, reasoning-focused models.

ACKNOWLEDGEMENTS

This work was conducted with the financial support of the Research Ireland Centre for Research Training in Digitally-Enhanced Reality (d-real) under Grant No. 18/CRT/6224. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission

REFERENCES

Bao, K., Zhang, J., Zhang, Y., Wang, W., Feng, F., and He, X. (2023). 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.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., and Raff, E. (2023). Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430.

Borisov, V., Seßler, K., Leemann, T., Pawelczyk, M., and Kasneci, G. (2022). Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., and Erlingsson (2021). Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.

Chen, L., Gao, C., Du, X., Luo, H., Jin, D., Li, Y., and Wang, M. (2024). Enhancing id-based recommendation with large language models. *arXiv preprint arXiv:2411.02041*.

Diao, S., Wang, P., Lin, Y., Pan, R., Liu, X., and Zhang, T. (2023). Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

Friedman, L., Ahuja, S., Allen, D., Tan, Z., Sidahmed, H., Long, C., Xie, J., Schubiner, G., Patel, A., and Lara, H. (2023). Leveraging large language models in conversational recommender systems. *arXiv preprint arXiv:2305.07961*.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., and Nabeshima (2020). The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Gao, Y., Sheng, T., Xiang, Y., Xiong, Y., Wang, H., and Zhang, J. (2023). Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.

Geng, S., Liu, S., Fu, Z., Ge, Y., and Zhang, Y. (2022). Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, pages 299–315.

He, R., Kang, W.-C., and McAuley, J. (2017). Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 161–169.

He, R. and McAuley, J. (2016). Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200.

Hou, Y., Zhang, J., Lin, Z., Lu, H., Xie, R., McAuley, J., and Zhao, W. X. (2024). Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer.

Jiang, C., Wang, J., Ma, W., Clarke, C. L., Wang, S., Wu, C., and Zhang, M. (2025). Beyond utility: Evaluating llm as recommender. In *Proceedings of the ACM on Web Conference 2025*, pages 3850–3862.

Kang, W.-C. and McAuley, J. (2018). Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE.

Kim, S., Kang, H., Choi, S., Kim, D., Yang, M., and Park, C. (2024). Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1395–1406.

Liao, J., Li, S., Yang, Z., Wu, J., Yuan, Y., Wang, X., and He, X. (2024). Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1785–1795.

Liu, J., Liu, C., Zhou, P., Ye, Q., Chong, D., Zhou, K., Xie, Y., Cao, Y., Wang, S., and You (2023). Llmrec: Benchmarking large language models on recommendation task. *arXiv preprint arXiv:2308.12241*.

Liu, J., Sun, Z., Feng, S., Chen, C., and Ong, Y.-S. (2024). Language model evolutionary algorithms for recommender systems: Benchmarks and algorithm comparisons. *arXiv preprint arXiv:2411.10697*.

Liu, Q., Zhu, J., Fan, L., Wang, K., Hu, H., Guo, W., Liu, Y., and Wu, X.-M. (2025). Benchmarking llms in recommendation tasks: A comparative evaluation with conventional recommenders. *arXiv preprint arXiv:2503.05493*.

Ni, J., Li, J., and McAuley, J. (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

Petrov, A. V. and Macdonald, C. (2023). Generative sequential recommendation with gptrec. *arXiv preprint arXiv:2306.11114*.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Shazeer, N. (2020). Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Wang, L. and Lim, E.-P. (2023). Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*.

Wu, X., Zhou, H., Shi, Y., Yao, W., Huang, X., and Liu, N. (2024). Could small language models serve as recommenders? towards data-centric cold-start recommendation. In *Proceedings of the ACM Web Conference 2024*, pages 3566–3575.

Xie, R., Wang, J., Huang, R., Zhang, M., Ge, R., Pei, J., Gong, N. Z., and Dhingra, B. (2024). Recall: Membership inference via relative conditional log-likelihoods. *arXiv preprint arXiv:2406.15968*.

Xu, S., Hua, W., and Zhang, Y. (2024). Openp5: An open-source platform for developing, training, and evaluating llm-based recommender systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 386–394.