360Brew : A Decoder-only Foundation Model for Ranking and Recommendation

360Brew Team
Foundation AI Technologies (FAIT), LinkedIn
Corresponding Authors: Maziar Sanjabi*, Hamed Firooz†

Abstract

Ranking and recommendation systems are the foundation for numerous online experiences, ranging from search results to personalized content delivery. These systems have evolved into complex, multilayered architectures that leverage vast datasets and often incorporate thousands of predictive models. The maintenance and enhancement of these models is a labor intensive process that requires extensive feature engineering. This approach not only exacerbates technical debt but also hampers innovation in extending these systems to emerging problem domains.

In this report, we present our research to address these challenges by utilizing a large foundation model with a textual interface for ranking and recommendation tasks. We illustrate several key advantages of our approach: (1) a single model can manage multiple predictive tasks involved in ranking and recommendation, (2) decoder models with textual interface due to their comprehension of reasoning capabilities, can generalize to new recommendation surfaces and out-of-domain problems in a zero-shot manner through simple prompts, thereby enabling product designers to engage with and iterate on them with ease, and (3) by employing natural language interfaces for task definitions and verbalizing member behaviors and their social connections, we eliminate the need for feature engineering and the maintenance of complex directed acyclic graphs (DAGs) of model dependencies. We introduce our research on our pre-production model, *360Brew* V1.0, developed by a small team of researchers and engineers over a 9-month period. *360Brew* is a 150B parameter, decoder-only model that has been trained and fine-tuned on LinkedIn's data and tasks. This model is capable of solving over 30 predictive tasks across various segments of the LinkedIn platform, achieving performance levels comparable to or exceeding those of current production systems based on offline metrics, without task-specific fine-tuning. Notably, each of these tasks is conventionally addressed by dedicated models that have been developed and maintained over multiple years by teams of a similar or larger size than our own.

The primary objective of *360Brew* is to enhance AI development productivity across LinkedIn by centralizing common modeling components into a foundational layer. This approach facilitates fundamental modifications, moderation, and governance at the foundational level, ensuring simultaneous benefits across all platforms. Additionally, developers are empowered to rapidly iterate on new objectives, data sources, and AI-driven search and recommendation products.

1 Introduction

Production-level Recommendation Systems (RS) are built to model member's intent and interest and match them with the items, such as contents, jobs, people, and provide a better experience for the members. These systems are at the heart of many internet applications, e.g., search, social networking, e-commerce, and many more [1, 2]. RS often consists of many layers and models. In Figure 1 we show a simplified version of such complex systems. On the one side are generally members whom the systems want to recommend jobs, posts, or people to connect with. The load on the system is defined in terms of queries per second, which is often dictated by the member count and their frequency of interactions. For example, if it is a search system, it is decided by how often people search on the platform. On the other side of the recommendation, members often need to be matched with items, e.g. jobs or posts.

*maz@linkedin.com
†hfirooz@linkedin.com

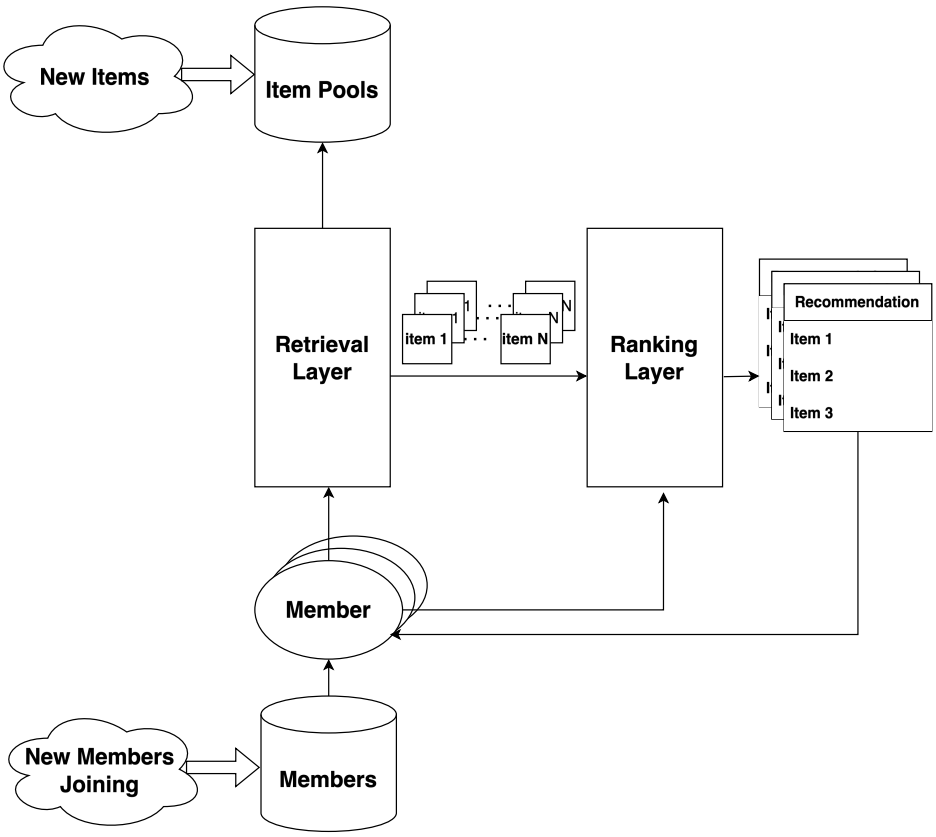

Figure 1: Overview of A Recommendation System (RS)

In most cases, the size of the item pool is much larger or at least equal to the size of the member pool. For example, the number of posts on LinkedIn is much larger than the number of LinkedIn members. Due to the large size of the item pool, the problem often needs to be solved in multiple stages, depicted here as retrieval and ranking layers[1]. In the retrieval layer, the idea is to quickly narrow down the number of possible items per member. In the ranking layer, the goal is to find a few best items among the retrieved items and rank them for the member to consume.

While both retrieval and ranking are trying to fundamentally solve the same problem (find the N best items for the member in a pool of K items), they have their own goals and challenges. For example, since retrieval is at the beginning of the RS funnel (large K and large N), it needs to have high recall. Moreover, due to large K, retrieval needs to be super-scalable and often uses simple member-item interaction architectures such as two tower models [3]. On the other hand, ranking happens after the retrieval and has a smaller number of potential items to filter (smaller K). This allows the usage of more complicated models, e.g. DCN [4]. At the same time, since the ranking model output is member-facing, it needs to have high precision (since N is small) and high recall to result in a good member experience.

In this work on our *360Brew* model V1.0, we mainly focus on ranking tasks since they need to have high precision and recall and are less bounded by computational constraints in practice (as mentioned above). We believe that it is possible to have the same model for retrieval (through embeddings, e.g. [5]) and ranking (through predictions), and it would be the focus of our next *360Brew* model releases.

1.1 Status Quo of RS Models

The retrieval and ranking layers in RS typically rely on ID based features for members and items. Essentially, these are large embedding tables where each embedding represents a (few) member(s)/item(s).

[1]The retrieval and ranking layers themselves can possibly consist of multiple sub-stages.

In addition to the ID based features, manually engineered features such as member/item interactions and item attributes are employed for richer representation learning. However, this process demands a perpetual process of improving and maintaining these models by large engineering teams.

In the rest of this section we discuss the challenges and active areas of research in RS modeling. Interestingly these challenges are mostly a direct result of using ID based and hand-crafted features.

- **Cold-start**: The RS models rely on currently available IDs and cannot generalize to new members and items without frequent re-training. This frequent and computationally heavy process is necessary for adapting to new items/members and varying member behaviors (see the following points). As a result, a large body of works have been dedicated to solving this issue through approaches such as content-based representations [6, 7].
- **Interactions**: Older RS models were primitive and required most of the cross features between member(s) and item(s) to be hand-crafted. With the popularity of deep neural networks (DNNs), architectures, such as DCN [4], which would allow automatic crossing of the item and member features became popular. More recently, there have been many works proposing more advanced architectures, e.g. transformers, for this purpose [8, 9].
- **Domain generalization**: Most of the RS models are super-specialized, through the use of ID based and specialized embedding features. Such models are trained on specific interactions from a specific domain and cannot generalize to new surfaces and tasks [10]. For example, they cannot go from feed recommendation to job recommendation (surface change), or start predicting member's interaction instead of member's ratings (task change).

In this work we introduce *360Brew* which is LinkedIn's venture project to build RS models based on modern and powerful LLMs that can naturally address the above-mentioned drawbacks of traditional RS models.

2 *360Brew*

2.1 Overview

Foundation models have achieved an unprecedented level of generality by leveraging scale and attention-based architectures [11, 12]. Recent advancements in decoder-only model architectures that use language as an input interface have demonstrated their capabilities in understanding, reasoning, and solving a variety of tasks beyond natural language processing, including graph understanding [13, 14, 15], robotic planning [16], and knowledge base management [17, 18], among others.

Recent research has explored the application of large pre-trained decoder-only models to member-item recommendation tasks [19, 10]. The emergent problem-solving and reasoning capabilities of these models allow them to process member and item information—such as member profiles and job or post descriptions—and assess their relevance with high accuracy. **This capability enables the models to exhibit improved performance on cold-start problems without additional modifications**.

Traditional ranking models often rely on hundreds of thousands of handcrafted features and complex architectures to capture high-order interactions among these features [4]. In contrast, the deep, multi-layer transformer architecture of LLMs can extract the necessary features directly from contextual information that is provided in text to address the task at hand [20]. **This approach eliminates the need for feature engineering at the item or member level, as all input is processed as text by the model**. Furthermore, LLMs are **adept at generalizing to different item types and interactions, enhancing their out-of-the-box ability to adapt to new items, surfaces and interaction patterns**. This property supports broad generalizability in new contexts.

In many practical use cases, traditional RS utilize text and content understanding models solely as feature generators to enhance ID-based models [6, 7]. However, our approach goes a step further. Instead of merely using these models to produce embeddings or features for the system, we aim to replace the entire model with an LLM that employs a natural language interface. **This approach is motivated by the well-documented ability of such models to identify and generalize patterns, as demonstrated by their reasoning capabilities [21] and few-shot problem-solving performance [22]**. As we discuss in the following sections, this capability is vital for highly fine-grained personalization in our RS.

In any recommendation task, each member, with their unique profile and history of interactions, can be viewed as a many-shot problem [23]. Consequently, when the ranking model is conditioned on the member's profile and interaction history, it can identify and generalize patterns that are highly personalized for that member, extending these patterns to future interactions. With this setup, we can reformulate a recommendation problem as solving tasks through **many-shot in-context learning (ICL)**. In other words, our foundation model is trained to approximate the following joint distribution for all members, interactions, and tasks:

$$P(\ m,\ (e_1,\ i_1), \cdots,\ (e_{N-1},\ i_{N-1}),\ (e_N,\ i_N)), \tag{1}$$

where u represents the member profile as text, and $(e_j, i_j),\ j = 1, \cdots, N$ are the set of historical items and interactions encoded as text for member m. Given this approximation of joint probability, the model can be used to estimate various marginal probabilities for different tasks:

$$P(i_t,\ i_{t+1}, \cdots |\ \text{Task Instruction},\ m,\ (e_1,\ i_1), \cdots,\ (e_{t-1},\ i_{t-1}),\ e_t,\ e_{t+1}, \cdots), \tag{2}$$

where "Task Instruction" represents the task, surface, or interaction description, and $e_t,\ e_{t+1}, \cdots$ are the new entities for which we want to predict the member's behavior. An example of a possible prompt is provided in Table 1. Note that this formulation is similar to the next token prediction approach used in training decoder-only LLMs. As a result, we can adapt this architecture for our foundation ranking model, which we call *360Brew* .

Instruction:
You are provided a member's profile and a set of jobs, their description, and interactions that the member had with the jobs. For each past job, the member has taken one of the following actions: applied, viewed, dismissed, or did not interact.
Your task is to analyze the job interaction data along with the member's profile to predict whether the member will apply, view, or dismiss a new job referred to as the "Question" job.

Note: Focus on skills, location, and years of experience more than other criteria. In your calculation, assign a 30% weight to the relevance between the member's profile and the job description, and a 70% weight to the member's historical activity.

Member Profile:
Current position: software engineer, current company: Google, Location: Sunnyvale, California.

Past job interaction data:
Member has applied to the following jobs: [Title: Software Engineer, Location: New York, Country: USA, Company: Meta, Description: ...]
Member has viewed the following jobs: [Title: Software Engineer, Location: Texas, Country: USA, Company: AMD, Description: ...]

Question:
Will the member apply to the following job: [Title: Software Engineer, Location: Seattle, Country: USA, Company: Apple, Description: ...]

Answer:
The member will apply

Table 1: A toy example of *360Brew* input for solving a job recommendation task. Note that the member and job data used in this example are entirely fictional.

As previously mentioned, by starting our modeling process with a pre-trained LLM architecture, we benefit from the model's inherent ability to understand items and members as separate entities. However, the model does not initially capture the relationships and interactions between them. By further training the model on these interactions, we enhance its few-shot capabilities, enabling it to

identify complex patterns in the member's history and extrapolate these to predict future interactions. This is achievable due to the architecture of large multi-layer transformers, which are adept at detecting and reasoning about intricate patterns [21, 22]. We will demonstrate this capability in the results section by showcasing the model's performance in solving a variety of in-domain and out-of-domain ranking and prediction tasks. These results are competitive with, or even surpass, those of current production models. *More details to come soon.*

Traditional large-scale industrial RS rely heavily on tens of thousands of handcrafted features, including features designed for different time horizons, such as multi-resolution count features [24]. Recent advancements in DNN architectures have sought to reduce the dependency on engineered features by enabling the models to discover these features from massive amounts of data [4, 25]. **Since the *360Brew* model relies solely on interaction history, it eliminates the need for hand-crafted, time-horizon-specific features, such as interactions per week, which are customary in traditional RS**.

As we will show in our results, an interesting byproduct of this approach is that more historical data improves the performance of the model. **This implies that we have replaced the problem of feature engineering with the more tractable challenge of scaling up architecture depth and context length, which primarily requires additional compute resources**. This highlights the advantage of LLM foundation models in optimizing human resource utilization by replacing the labor-intensive (and non-scalable) process of feature engineering with a compute-driven approach that relies on capital expenditure and is inherently scalable.

2.2 Results

In this section, we present the results from developing the *360Brew* V1.0 model. The *360Brew* model V1.0 is built on top of Mixtral 8x22 pre-trained MoE [26] architecture[2]. We further train it using a combination of LinkedIn raw entity data—such as member profiles, job descriptions, and LinkedIn posts—and interaction data (including member job applications) across 5+ surfaces on LinkedIn. *More details to come soon.*

Finally, we apply the model to 30+ tasks across 8+ surfaces. Notably, not all tasks are in-domain, as some surfaces and tasks are outside the training data that the model has seen. Specifically, we categorize the tasks as follows:

- **T1** (in-domain): Tasks where recommendation data from past periods is used in training the model. There is at least a one-month gap between the T1 data used in training and the benchmark dataset. This means the data is subject to distribution shift, which is very common in recommendation systems, but the model has seen member behavior in the past for these tasks.
- **T2** (out-of-domain): Tasks and surfaces that are not part of the training data. In these tasks, the model is evaluated on data from domains or recommendation use cases that it has not encountered during training, representing an out-of-domain distribution.

Our model is flexible and can predict different type of outputs based on the prompt. However, for simplicity and ease of scaling, the majority of traditional large industry ranking models are framed as binary tasks, such as $P(\text{like})$. Therefore, we also adapt our prompts so that the model generates binary predictions when answering questions about whether a member likes a post, allowing for direct comparison with these models. We use the logits for these tokens to obtain scores and compute metrics such as AUC, etc.

2.3 Data Scaling

As mentioned earlier, one of the most compelling benefits of building foundation models for ranking is the reduced development time. The way that this goal is achieved is by training a single large model on a large amount of data [27, 11]. In Figure 2, we show that as we scaled our processed data by over 3x[3], the model's performance on T1 tasks improved significantly compared to the baselines. Notably, the baseline models were developed through multi-year investments, each requiring dedicated teams of engineers for maintenance and ongoing development. In contrast, these data scalings were achieved

[2] *More details to come soon.*
[3] *More details to come soon.*

within a nine-month period by a relatively small team. Furthermore, it is important to highlight that the slope of improvement with increased data size does not appear to be flattening. This indicates that our model continues to identify useful patterns as more data is added. As such, we anticipate further improvements by incorporating additional data in the future.

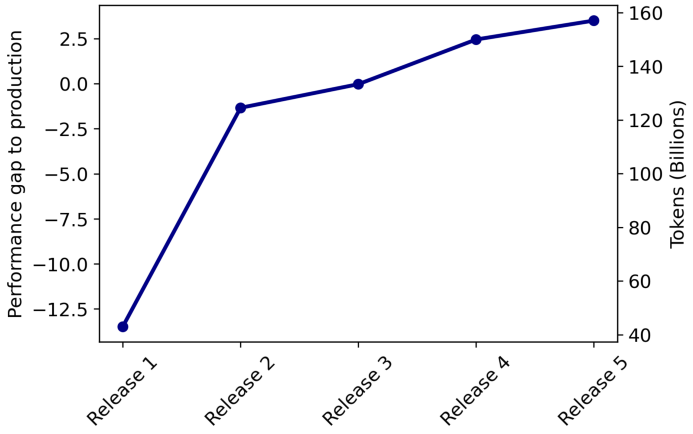

Figure 2: Effects of data size (in tokens) on the average performance of the model compared to baselines across five releases over a 9-month period.

2.4 Model Scaling

Recent works have shown increasing the size of the LLMs improves their capabilities in understanding the context and performing tasks [11]. In Figure 3 we show that the same holds in RS use-cases and *360Brew* models get better as we increase the size of the model (by using larger and more powerful pre-trained architectures). This shows that there is an added value in modeling more complex relationships or reasoning by using larger model architectures in RS use-cases.

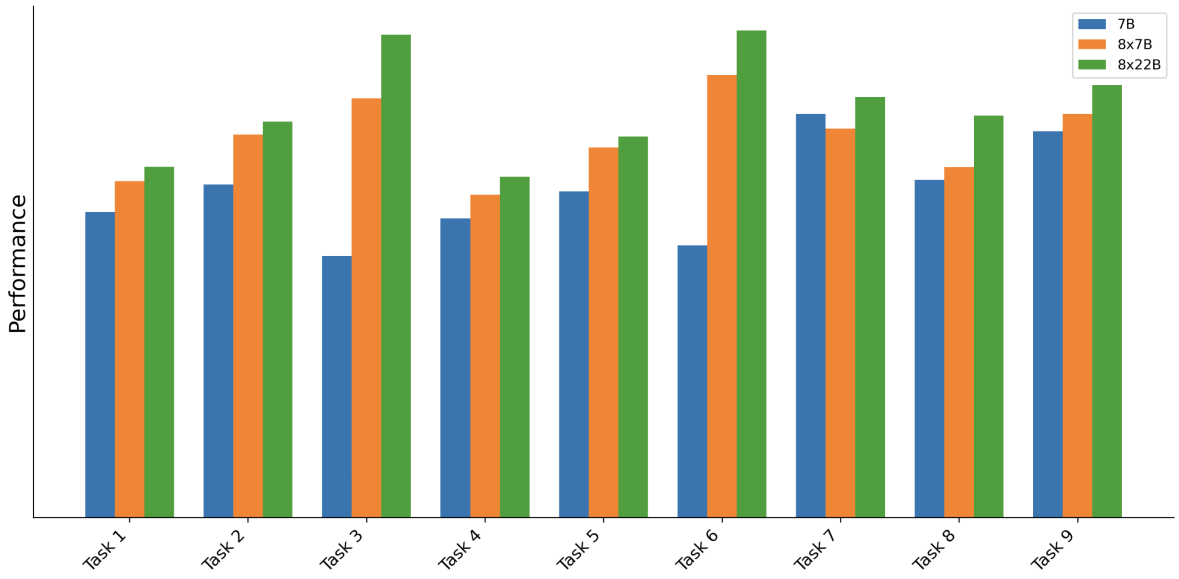

Figure 3: Effects of model size (in parameters) on the performance across different tasks.

2.5 History Scaling

As mentioned above, one advantage of *360Brew* model is that it does not require manual feature engineering. In Figure 4, we show that *360Brew* model's performance gets better as we increase the history by increasing the max context length. The take-home message here is that improving processed history (by using more compute) could be an effective way of improving the performance of the RS. It is worth noting that the story here is a bit more nuanced than other scaling laws since the performance of the model could be affected by the underlying model's limitation on its context length. For more details see *More details to come soon.*We believe that the technical and modeling limitations on the context length will be alleviated with more advances in pre-trained LLM architectures. Based on our results, such advancements could directly benefit the *360Brew* model.

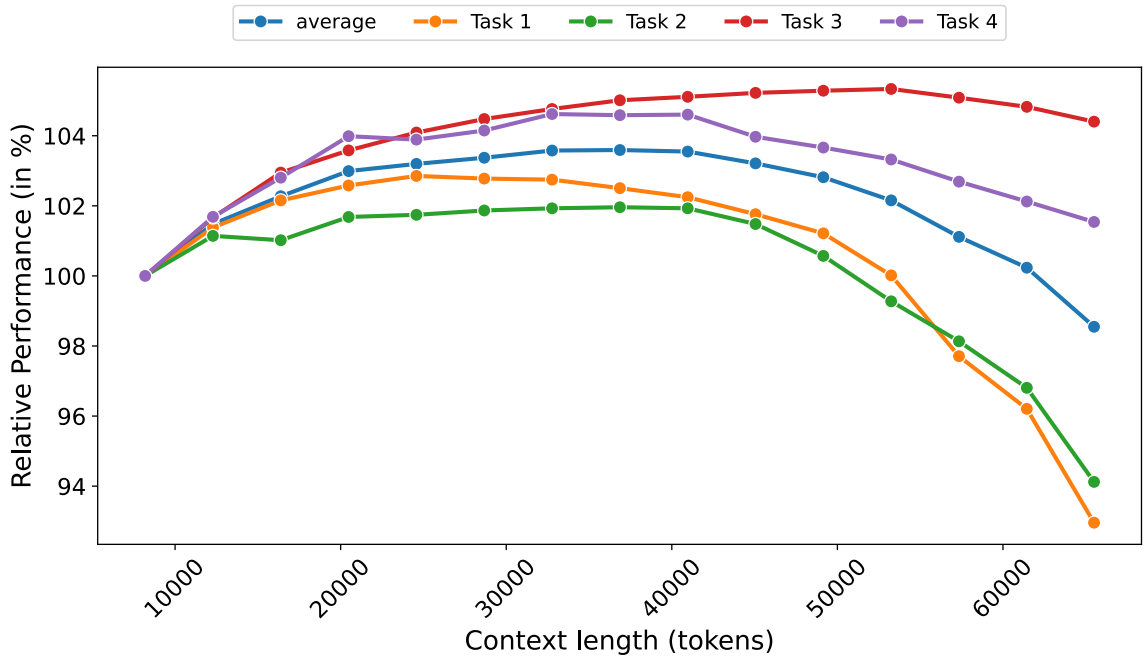

Figure 4: Effects of max context length (in tokens) on the performance of *360Brew* . The performance at each maximum context length is normalized by the performance at 8K maximum context length.

2.6 Generalization

2.6.1 Out of domain

In Figure 5 we show that the *360Brew* model can generalize to out-of-domain tasks and surfaces, and achieves performance similar to or better than the production model.

2.6.2 Cold-start

In Figure 6 we show the gap between the *360Brew* model and the production model as max number of historical member interactions decreases from 100 to 5. As shown in the plot, the performance gap between the two models is the largest when member has few available interactions, which shows that *360Brew* model has a greater margin over the production model for members with fewer interactions. These are usually called cold start members since they do not have many interactions.

2.6.3 Temporal

One major downside of the current recommendation systems is that they do not generalize well over time, and the relationships that they have extracted might be irrelevant to the data distribution shifts.

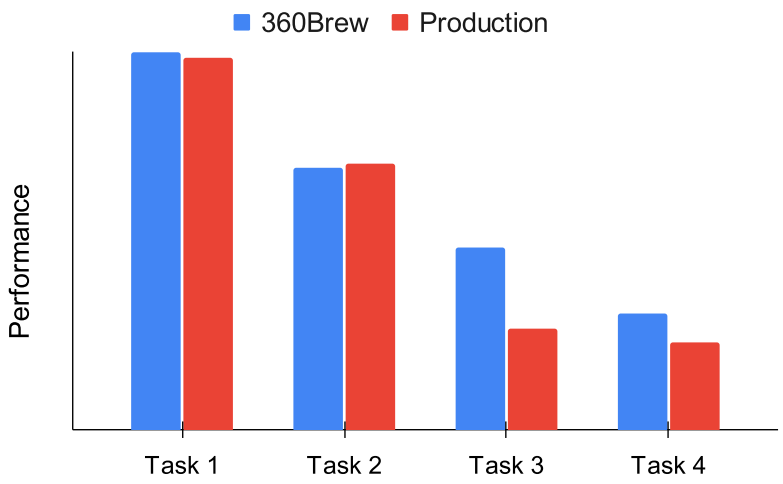

Figure 5: Performance on 4 T2 tasks across 4 surfaces which were not part of *360Brew* training.

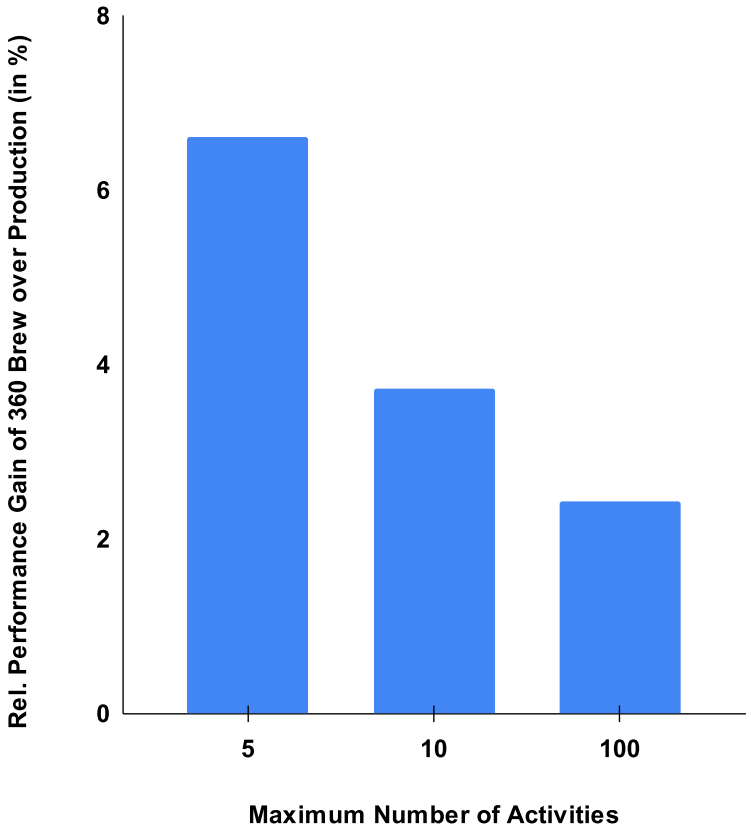

Figure 6: Relative performance gap between *360Brew* and production model as a function of max number of member interactions. For the members with a lower interaction count, *360Brew* performs better and has a larger gap with the baseline.

This results in frequent re-training of the model which increases the maintenance cost, development time, and complexity of the systems. In Figure 7, we benchmark the performance of the *360Brew* model on different test data that lie within different time frames (in the future) from the training data. We do the same with the baseline models and show that the performance of the *360Brew* model is less affected by time. This means that using *360Brew* could potentially lead to more developer efficiency and less maintenance and technical debt as we do not need to update the model so frequently. This is possible since *360Brew* model can use ICL to adjust its answers based on the member behavior it sees in its context. *More details to come soon.*

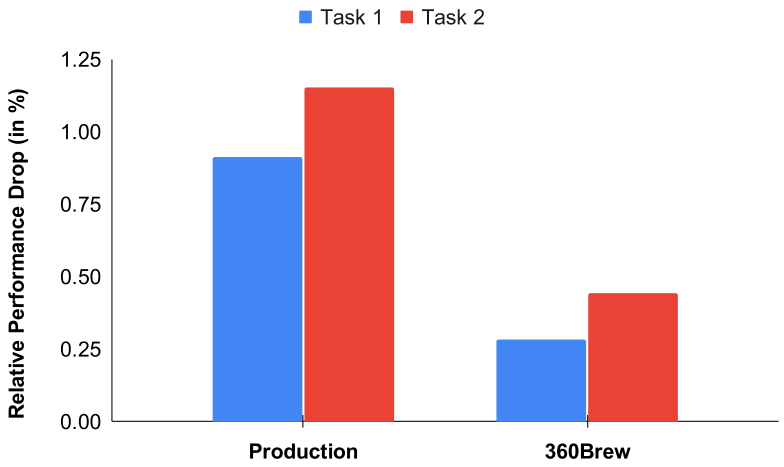

Figure 7: Performance of *360Brew* model compared to baselines as the test data gets temporally farther from the training data.

References

[1] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM computing surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[2] D. Roy and M. Dutta, "A systematic review and research perspective on recommender systems," *Journal of Big Data*, vol. 9, no. 1, p. 59, 2022.

[3] P. Li, S. A. M. Noah, and H. M. Sarim, "A survey on deep neural networks in collaborative filtering recommendation systems," *arXiv preprint arXiv:2412.01378*, 2024.

[4] R. Wang, R. Shivanna, D. Cheng, S. Jain, D. Lin, L. Hong, and E. Chi, "Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems," in *Proceedings of the web conference 2021*, 2021, pp. 1785–1797.

[5] A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sinha, V. Ramanujan, W. Howard-Snyder, K. Chen, S. Kakade, P. Jain *et al.*, "Matryoshka representation learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 233–30 249, 2022.

[6] X. Ren, W. Wei, L. Xia, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Representation learning with large language models for recommendation," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 3464–3475.

[7] J. Chen, L. Chi, B. Peng, and Z. Yuan, "Hllm: Enhancing sequential recommendations via hierarchical large language models for item and user modeling," *arXiv preprint arXiv:2409.12740*, 2024.

[8] J. Zhai, L. Liao, X. Liu, Y. Wang, R. Li, X. Cao, L. Gao, Z. Gong, F. Gu, M. He *et al.*, "Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations," *arXiv preprint arXiv:2402.17152*, 2024.

[9] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, pp. 197–206.

[10] J. Li, M. Wang, J. Li, J. Fu, X. Shen, J. Shang, and J. McAuley, "Text is all you need: Learning language representations for sequential recommendation," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 1258–1267.

[11] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[12] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[13] B. Perozzi, B. Fatemi, D. Zelle, A. Tsitsulin, M. Kazemi, R. Al-Rfou, and J. Halcrow, "Let your graph do the talking: Encoding structured data for llms," *arXiv preprint arXiv:2402.05862*, 2024.

[14] H. Firooz, M. Sanjabi, W. Jiang, and X. Zhai, "Lost-in-distance: Impact of contextual proximity on llm performance in graph tasks," *arXiv preprint arXiv:2410.01985*, 2024.

[15] B. Fatemi, J. Halcrow, and B. Perozzi, "Talk like a graph: Encoding graphs for large language models," in *The Twelfth International Conference on Learning Representations*, 2024.

[16] J. Andreas, "Language models as agent models," *arXiv preprint arXiv:2212.01681*, 2022.

[17] B. AlKhamissi, M. Li, A. Celikyilmaz, M. Diab, and M. Ghazvininejad, "A review on language models as knowledge bases," *arXiv preprint arXiv:2204.06031*, 2022.

[18] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" *arXiv preprint arXiv:1909.01066*, 2019.

[19] J. Zhang, R. Xie, Y. Hou, W. X. Zhao, L. Lin, and J.-R. Wen, "Recommendation as instruction following: A large language model empowered recommendation approach," *arXiv preprint arXiv:2305.07001*, 2023.

[20] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5)," in *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022, pp. 299–315.

[21] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[22] Y. Bai, F. Chen, H. Wang, C. Xiong, and S. Mei, "Transformers as statisticians: provable in-context learning with in-context algorithm selection," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023, pp. 57 125–57 211.

[23] R. Agarwal, A. Singh, L. M. Zhang, B. Bohnet, S. Chan, A. Anand, Z. Abbas, A. Nova, J. D. Co-Reyes, E. Chu *et al.*, "Many-shot in-context learning," *arXiv preprint arXiv:2404.11018*, 2024.

[24] R. Anil, S. Gadanho, D. Huang, N. Jacob, Z. Li, D. Lin, T. Phillips, C. Pop, K. Regan, G. I. Shamir *et al.*, "On the factory floor: Ml engineering for industrial-scale ads recommendation models," *arXiv preprint arXiv:2209.05310*, 2022.

[25] J. Zhai, L. Liao, X. Liu, Y. Wang, R. Li, X. Cao, L. Gao, Z. Gong, F. Gu, M. He *et al.*, "Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations," *arXiv preprint arXiv:2402.17152*, 2024.

[26] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[27] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. Casas, L. Hendricks, J. Welbl, A. Clark *et al.*, "Training compute-optimal large language models. arxiv 2022," *arXiv preprint arXiv:2203.15556*, vol. 10, 2022.

A *360Brew* team members (alphabetical)

Adrian Englhardt, Aman Gupta, Ben Levine, Dre Olgiati, Gungor Polatkan, Hamed Firooz, Iuliia Melnychuk, Karthik Ramgopal, Kirill Talanine, Kutta Srinivasan, Luke Simon, Maziar Sanjabi, Natesh Sivasubramoniapillai, Necip Fazil Ayan, Qingquan Song, Samira Sriram, Souvik Ghosh, Tao Song, Vignesh Kothapalli, Xiaoling Zhai, Ya Xu[4] , Yu Wang, and Yun Dai

[4]Now at Google DeepMind