Evaluating Auto-complete Ranking for Diversity and Relevance

Sonali Singh[1][0000−0002−2706−2442], Sachin Farfade[2][0000−0002−1619−9873], and Prakash Mandayam Comar[3][0000−0003−0817−7938]

[1] Amazon, ssonl@amazon.com
[2] Amazon, sfarfade@amazon.com
[3] Amazon, prakasc@amazon.com

Abstract. Traditional Query Auto-completion (QAC) systems optimise for query relevance based on past user interactions. This approach excels at surfacing frequently searched queries, but ensuring a diverse range of suggestions and incorporating new products or trends often requires post-processing heuristics. This limitation stems from relying on user search logs, which may not fully capture the evolving product landscape. This paper presents a comparison of traditional state-of-the-art (SOTA) methods with Large Language Models (LLMs) for query auto-completion. The LLMs, with their ability to understand language and product information, can generate a wider range of relevant and diverse suggestions, encompassing the entire product catalog and potentially including entirely new queries. Here, we study the trade-off between latency, relevance and comprehensiveness in the QAC systems. Our experiments on real world data show that the LLM based QAC system offers a significant 38% boost in diversity with no significant change in relevance metrics. However, their high compute and memory demands make them less suitable for real-time applications. We introduce a heuristic approach that integrates the strengths of existing methods with the power of LLMs. This combined approach has been shown to yield a measurable 0.13% increase in sales revenue in live A/B testing.

Keywords: query auto-completion, large language models, relevance, diversity, latency

1 Introduction

The QAC (Query Auto-completion) system [9] acts as an essential entry point for e-commerce sites, assisting millions of users in quickly finalizing their search queries while also uncovering new and trending product-related keywords they hadn't encountered before. These QAC systems typically employ ranking models trained to rank popular queries, matching the given prefix using Learning to Rank (LTR) [10] algorithms. The process begins by identifying candidate queries matching the prefix, followed by ranking the top candidates to maximize the query acceptance rate.

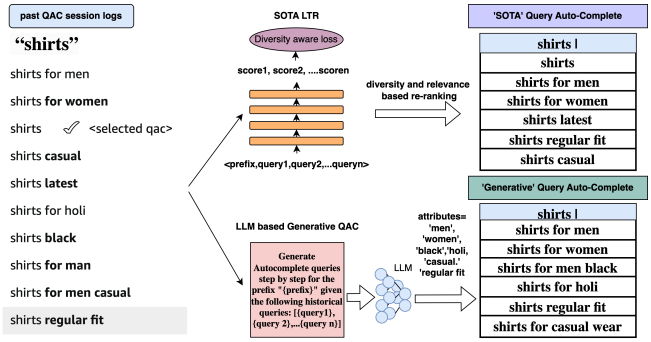

Fig. 1. Schematic illustrating enhancing QAC experience balancing relevance and diversity using SOTA LTR losses vs Generative QAC by prompting LLMs.

Traditional QAC systems leverage Learning-to-Rank (LTR) algorithms trained with various loss functions:- pointwise, pairwise and listwise [10, 32] to optimize relevance based on user search logs. While LTR excels at ranking frequently searched queries, it struggles to capture the full product range. New additions like "vacation dresses" might not surface prominently unless explicitly searched. Large Language Models (LLMs) offer a compelling alternative. By understanding both language and product information, LLMs have the ability to generate a broader array of relevant and varied suggestions for every prefix, covering the entire product catalog. This eliminates the need for separate diversity algorithms like Maximal Marginal Relevance [11] or Determinantal Point Processes [17], which are often employed with LTR to improve suggestion variety. Furthermore, LLMs possess the capability to handle novel query possibilities more effectively by generating new queries through the employment of effective prompting techniques. This shift from LTR to LLM-based approaches holds promise for a more comprehensive and dynamic QAC system, potentially leading to a more satisfying user experience. Large Language Models (LLMs) offer a significant improvement over traditional QAC systems that rely on user search logs. Trained on massive amounts of text data, LLMs excel at generating well-formed queries, unlike the potentially misspelled or incomplete queries often found in search logs. Beyond synthetic query generation, LLMs can also be leveraged to rank these suggestions. Efficacy of LLM as zero-shot rankers is demonstrated for both document ranking and recommendation tasks [14] through the utilization of pairwise-prompting techniques [22]. However, the document and item ranking contexts are typically broader than e-commerce QAC systems. The effectiveness of LLMs as rankers with minimal input, such as just a prefix or a few characters, remains unexplored. To address these concerns, we propose a comparative analysis of LLM performance against state-of-the-art (SOTA) neural network (NN) based ranking algorithms as illustrated in Figure 1. This analysis will help us understand the strengths and limitations of LLMs in the specific context of e-commerce QAC ranking.

We leverage internal e-commerce search logs and publicly available AOL search logs [19] to conduct a series of experiments aimed at addressing several critical questions within the domain of query auto-completion (QAC) ranking:

- **CQ1**: Can Language Models based ranking methods outperform traditional prefix to query ranking strategies in e-commerce QAC?
- **CQ2**: Can LLMs effectively tackle diversity issues through prompting, or is a diversity-aware loss indispensable in ranking tasks?
- **CQ3**: Can the generative nature of LLM be successfully adopted in real-time auto-complete application?
- **CQ4**: Does LLM based understanding of prefixes and search queries bring value to QAC system?

Our observations reveal that, although LLMs demonstrate exceptional performance in diversifying QAC, they only marginally surpass the relevance benchmark. Moreover, it fails to meet the latency requirements of large e-commerce store. Hence, we propose an approach to leverage the inherent generative capabilities of the LLM methodology to create high-quality, diverse candidate queries which are then merged into the organic candidate list and subsequently ranked by the SOTA NN ranking model. We empirically demonstrate that this approach leads to a significant reduction in redundant and low-quality queries in QAC which lead to 0.13% revenue improvement in online A/B test.

2 Related Work

Query Auto-completion (QAC) systems encompass two approaches: retrieving and ranking historical customer queries that match the user's prefix, or employing a generative approach to generate a list of completions. Traditionally, retrieval and ranking relied on heuristics, with Most Popular Completion algorithms being a prevalent choice [3]. Subsequently, learning-based methods emerged, often incorporating time-sensitive signals and user feedback to enhance the ranking process [8, 15]. Generative heuristic approaches, on the other hand, employed techniques such as templates and n-grams [26, 4] for QAC. Later advancements embraced Generative Neural Language Models (LMs) [16, 28] to synthesize QAC suggestions. Cai et al. [9] provide a comprehensive survey of these traditional QAC approaches.

Our research explores, compares, and contrasts traditional ranking approaches for QAC with Large Language Models (LLMs) that can be strategically employed to enhance the quality of suggestions in QAC within the e-commerce domain without compromising query clicks and downstream sales. Initially, we assess the performance of traditional ranking approaches by employing well-established learning-to-rank pairwise and listwise objective functions [10, 7, 12] to optimize for relevance. We also benchmark on state-of-the-art (SOTA) diversity-enhancing techniques in the traditional ranking setting by introducing additional objective functions within pairwise loss [24] or by directly optimizing diversity aware listwise losses [31]. [24] proposes improving QAC quality through multi-objective

ranking, boosting navigational queries via pairwise ranking. Additionally, we investigate the application of diversification-aware listwise techniques in QAC, drawing inspiration from diversifying web search results. We adapt this through a listwise Learning to Rank (LTR) framework [10] to score queries, incorporating a query interaction layer inspired by the Document Interaction Network (DIN) [20] to generate higher-order features for queries in the list. Benchmarks and investigations into effective neural architectures and loss functions for Learning to Rank are presented in [23]. Building upon these insights, we utilize the optimization metric as a loss function, as suggested by [6]. Furthermore, we explore the integration of LLMs as rankers within QAC systems. The utilization of LLMs as rankers has been examined in the context of document retrieval ranking and recommendation tasks. Initial efforts focused on fine-tuning pre-trained language models for supervised ranking tasks with ranking loss [18, 33]. With the recent notable performance of LLMs in Natural Language Processing (NLP) tasks, there has been growing interest in assessing their zero-shot performance in retrieval and ranking capabilities. [25] investigates the use of ChatGPT as a re-ranking agent. Additionally, [14] formalizes the recommendation task as a conditional ranking task, considering sequential historical interactions as conditions. The items retrieved by candidate generation models are treated as candidates and further ranked by constructing natural language prompts that incorporate historical interactions and candidates. [22] introduces the Pairwise Prompting technique (PRP), which reduces the complexity of employing LLMs as a ranking agent and demonstrates competitive performance using open-sourced LLMs. Furthermore, [34] compares all existing LLM-based zero-shot ranking approaches and introduces the novel Setwise approach which improves the effectiveness of the listwise prompting approach.

3 QAC Problem formulation

QAC can be approached as either a Learning-to-Rank (LTR) task or a generative task. We delineate both formulations as follows:

QAC as a LTR Problem: QAC can be conceptualized as an LTR task where, for a given prefix p, the initial step involves retrieving a set of top-matched candidates L. Subsequently, the objective is to learn a scoring mechanism s_k for each completion $k \in L$ with the highest score assigned to the most relevant query. These scores are acquired through training on appropriate ranking losses, optimizing for relevance and other pertinent business criteria. The determination of completion relevance typically hinges on user acceptance, specifically the likelihood of a user accepting or clicking on one of the queries in the QAC.

QAC as a Generative Problem: In this formulation, given a prefix p and a context denoted by C, where C could represent a list of historical queries L, a set of instructions, or other relevant information, the objective is to generate a ranked list of the top K suggestions that align with the provided prefix.

4 SOTA rankers with ranking loss

Pairwise and listwise ranking methodologies represent widely embraced strategies within the framework of LTR methods. We conduct experiments applying these methodologies to QAC with the following details:

Pairwise Ranking: Pairwise ranking losses, such as RankNet, LambdaRank, and LambdaMART [7], are widely adopted for pairwise ranking tasks. In our experiments, we choose to optimize a pairwise cross-entropy loss based on LambdaRank [7], as it is easily adaptable to different scoring architectures and business needs, with the goal of enhancing customer behavior.

Listwise Ranking: While pairwise loss remains a popular choice in designing ranking systems due to its simpler architectures and fast inference, it lacks the context of other queries in the list, which can be a limitation. Listwise losses, such as Softmax loss, ListMLE, ListNET, and approxNDCG [30, 27, 5], tend to mitigate these issues.

Handling diversity: In the context of e-commerce QAC, we define diversity as the presence of varied product types, categories, and attributes (navigational terms) within the queries. Conversely, for generic web search QAC, diversity often implies the inclusion of varied topics or subtopics. We leverage distinct objective functions that optimize for relevance and diversity, employing pairwise and listwise loss mechanisms. For the pairwise loss, we integrate the diversification strategy proposed by [24], which introduces a query quality (QQ) objective.

Recent works on diversification directly optimize diversification metrics [31] to introduce diversity in listwise loss. We investigate optimizing an approximate version of the diversity metric αNDCG through a listwise approach.

5 LLMs as rankers

LLMs have demonstrated effective zero-shot ranking performance in document ranking benchmarks [14, 25]. However, their performance in QAC ranking tasks remains unexplored. E-commerce QAC are constructed to be concise, often lacking grammatical correctness. For instance, a completion such as "men's running shoes" may lack clarity, whereas a more meaningful sentence could be "I want to buy running shoes for men." LLMs may not perform optimally on such proprietary search query datasets of companies. We explore leveraging LLMs as rankers by employing both generative and ranking loss functions, aiming to mitigate these challenges and enhance their performance in QAC ranking tasks.

5.1 Generative Loss

LLMs are trained to maximize log-likelihood using cross-entropy loss for next-word prediction tasks. A straight forward approach to leverage LLMs for ranking is to formulate the ranking task as a generative sequence-to-sequence problem. Various prompting styles can be used for fine-tuning sequence-to-sequence LLM based ranking for QAC as explained below (Figure 2).

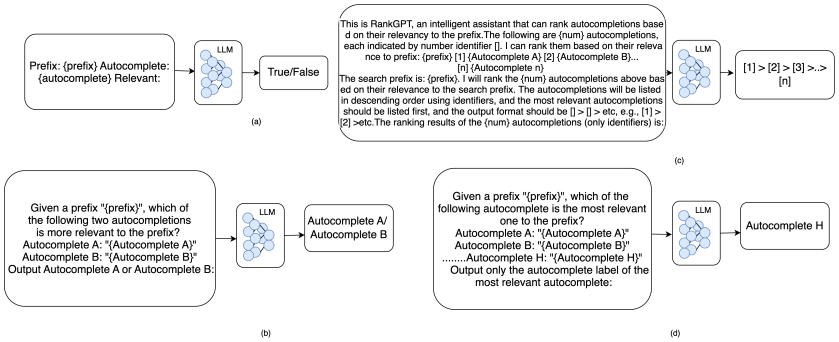

Fig. 2. Prompting techniques for ranking QAC using sequence-to-sequence LLMs: (a) Pointwise prompting for prefix-candidate relevance; (b) Pairwise prompting to generate label for more relevant candidate; (c) Listwise prompting to re-rank and generate sorted indices; (d) Setwise prompting to output most relevant candidate label from a set.

Pointwise Prompting: In this approach, we employ pointwise training of sequences similar to [18], where for a given prefix p and completion c, the large language model (LLM) is prompted to generate either "true" or "false" as output, indicating whether the completion c is relevant to the query prefix p. These generated tokens are further converted to relevance scores by applying the softmax function over the probabilities of generating the "true" or "false" token, as indicated by the following equation:

$$\text{score}(c|p) = \frac{e^{P_\theta(\text{“true”}|p,c)}}{e^{P_\theta(\text{“true”}|p,c)} + e^{P_\theta(\text{“false”}|p,c)}} \tag{1}$$

Here, $P_\theta(\text{“true”}|p,c)$ and $P_\theta(\text{“false”}|p,c)$ represent the probabilities of the LLM generating the "true" and "false" tokens, respectively, given the prefix p and completion c, parameterized by the LLM parameters θ.

Pairwise Prompting: The pairwise prompting [22] technique has emerged as one of the most effective methods for document ranking. In this approach, the model learns to rank the more relevant candidate given a pair of candidates (c_i, c_j) and a prefix p. The pairwise P_θ^{PRP} prompting involves prompting the large language model (LLM) twice with an ordering and a swapped ordering of the candidate pairs to obtain rankings (r_i, r_j). If the output remains consistent, scores are assigned based on the following label:

$$score(r_i > r_j|p, c_i, c_j) = \frac{1}{2}\left[P_\theta^{PRP}(p, c_i, c_j) + 1 - P_\theta^{PRP}(p, c_j, c_i)\right] \tag{2}$$

where $P_\theta^{PRP}(p, c_i, c_j)$ is an indicator function that takes the value 1 if the LLM outputs the label for c_i as more relevant, and 0 otherwise. Sorting techniques such as heap sort and bubble sort having $\mathcal{O}(n \log n)$ complexity are used at the time of inference to rank the entire list of items.

Listwise Prompting: The listwise prompting [21] approach ranks candidates by considering the entire context of the list. For short QAC queries, LLMs can be fine-tuned to generate a ranked list of suggestions. Instead of generating complete sequences, an alternative approach generates labels for sequences in ranked order to avoid queries outside the candidate list. Given a list of candidates with labels [1], [2], [3]...[n], the listwise prompting given to the LLM $P_\theta^{LIST}(p, c_1, c_2, ...c_n)$ generates a permutation according to the following equation:

$$P_\theta^{LIST}(p, c_1, c_2, ..., c_n) = [1] > [2] > [3] > [4].. > [n] \tag{3}$$

To accommodate lists that exceed the context window size, the list is partitioned into chunks, and the most relevant candidate is inserted into each chunk. During inference, a sliding window strategy [34] is employed.

Setwise Prompting: The setwise prompting technique [34] was introduced to achieve the power of efficiency of pairwise prompting and retain the advantage of the listwise technique that provides a larger context. As the name indicates, setwise prompting involves comparing a set of items at a time, which sits in between the pairwise and listwise approaches. This suitably eliminates the issue of the list of items being unable to fit in the context window and also the added complexity of pairwise inference.

5.2 Ranking loss

LLMs can be leveraged as feature generators with traditional ranking loss. Here, we plug in encoder of LLM as the feature generator network and add the traditional score generator network to generate the score per candidate. This combination of network is then trained on listwise loss based on the scores predicted by the network. Note that this method tends to be even more computationally complex as each candidate is scored individually unlike pairwise or listwise prompting techniques. Another variant to this approach is, instead of using additional network on top of LLM to generate the score, directly compute the probability of sequence (candidate) generation using teacher forcing [29] at the time of decoding and use these probability as scores in the ranking loss.

6 Diversification in QAC using LLMs

LLMs with their world knowledge are expected to understand the issue of diversity if prompted with suitable instructions. This translates the difficulty to comprehend the task of diversity from equations to simple english prompts illustrated in Figure 3.

Chain of thought [COT] prompting: A logical thought process to generate relevant and diverse QAC for a prefix is to rank the most popular autocomplete with a popular product type followed by the next popular product type or attribute. Hence, we design a prompt with instruction to order the candidates along-with mentioning attributes in the candidate indicating the navigational

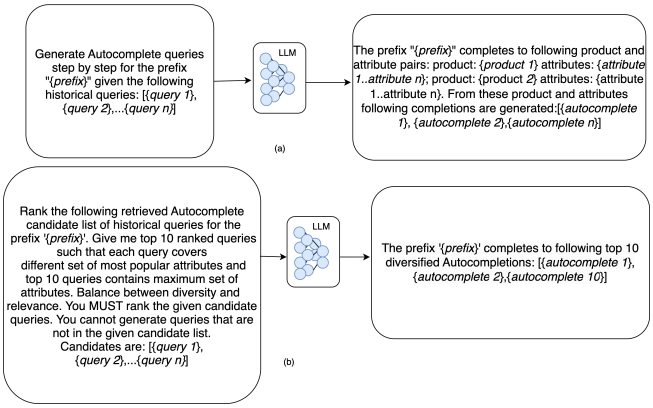

Fig. 3. Diversity-aware prompts for fine-tuning LLMs. (a) Chain of thought prompting to generate diverse attributes and auto-completions given prefix and corresponding candidates. (b) Prompt based instruction fine-tuning where the LLM is instructed to re-rank candidates balancing relevance and diversity.

component of the candidate.

Instruction Fine-tuning [Instruct-FT]: In this approach, we design a prompt instructing the model to rank the candidates while balancing relevance and diversity. We employ a heuristic algorithm to generate training samples, where we first sort the items by their popularity, as labeled from logs, and then sequentially include the next candidate only if it adds either a new attribute or product type. This carefully curated training data is then utilized for fine-tuning the model.

Diversity aware loss: Instead of using generative loss, we use diversity aware loss for training and use LLM as a feature generator with a traditional ranking network as score generator. Jointly, we fine-tune the complete network on both relevance and diversity.

7 Evaluation and Results

7.1 Experiment Setup

Datasets:

E-commerce dataset:Anonymized logs from an e-commerce store are utilized for generating training samples. A week's worth of customer logged data is used to create a mapping of prefixes to candidates by compiling the top 100 clicked candidates for each prefix. Each row of the evaluation sample contains a prefix, selected keyword, 100 candidates, past searches, device type, and other relevant details. Prefixes with fewer than 10 candidates are excluded from the evaluation. Additionally, prefixes without a click relevance label (i.e., sessions where no auto-complete candidate was selected) from the logs are omitted. The

data is randomly down-sampled to obtain 80k prefixes for training and 30k prefixes for testing, selected from different days within a month. Features used for training all models with ranking losses are calculated with approach mentioned in [24]. For training the LLMs, numerical features are discarded, and the prefix and candidates are used as context, which is fine-tuned with the selected candidate as the label.

AOL search logs: We utilize publicly available query topic analysis data [1], derived from AOL search logs [19]. This dataset contains the top 1000 user search queries with a primary and secondary topic, along with associated scores with a total of 318,023 queries. In the absence of prefix for a query, we randomly select a prefix for each query, following a strategy similar to [16]. Each searched query is treated as the clicked query, and a list of the top 100 clicked queries starting with the same prefix from these logs is added as candidates. 90k prefixes are randomly sampled for training, and 30k prefixes are set aside for testing. We create a query relevance matrix, containing 30 topics, by combining click-based relevance and aggregating the top 29 most frequently occurring topics per prefix-candidate list, used for both loss computation and evaluation. We utilize a 384-dimensional embedding from the Sentence Transformer model "multi-qa-MiniLM-L6-cos-v1" [2], designed for semantic search, as a feature for each candidate. Our feature set also includes aggregated click information, cosine similarity of the query embedding with the two previous queries within a 300-second timeframe, the ratio of prefix to query length, a binary feature indicating the presence of recent past two searches, and the time difference between the ongoing search and the two prior searches.

Model Architecture: We employ DeepPLTR [32], a RankNet-based Siamese NN model, for pairwise loss and moDPLTR [24] for diversity-aware pairwise loss. ListLTR, trained on softmax loss, has a neural ranker architecture with interaction layers [23]. DiALTR, the diversity-aware version for listwise ranking, retains ListLTR's architecture with a change in loss. For supervised LLM fine-tuning with ranking loss (referred as ListFlanT5-XLLTR), we use FLAN-T5-XL's [29] encoder as the embedding generator, retaining ListLTR's scoring architecture. We also train ListFlanT5-XLRankGen model with a combination of ranking (softmax) and generative loss, where ranking loss scores are computed by candidate generation probability and has architecture same as FLAN-T5-XL. For all prompt-based fine-tuning, we use FLAN-T5-XL with LoRA [13] for consistency.

Evaluation Metric:We utilize MRR (Mean Reciprocal Rank) and NDCG (Normalized Discounted Cumulative Gain) to evaluate click-based relevance, and αNDCG to assess diversity. MRR and NDCG are computed on the top 10 candidates per prefix, using the clicked candidate as the true label. αNDCG is computed with the top 29 most frequent topics as true labels along-with click labels obtained from logs for each prefix and candidate list. For each evaluation row, the topics are generated by mining attributes from e-commerce QAC candidates using an internal attribute tagger tool, and binary relevance labels are assigned if the topic/attribute is present within the candidate.

7.2 Results

Table 1. Relevance metrics comparison of traditional and LLM-based ranking networks relative to DeepPLTR baseline. Relative lifts on e-commerce data; absolute numbers on public AOL dataset.

Model	E-commerce Dataset		AOL Dataset			
	MRR	NDCG@10	MRR	NDCG@10	Latency	Parameters
Ranking loss						
DeepPLTR	-	-	0.384	0.438	2ms	50k
ListLTR	+9.31%	+5.28%	0.417	0.469	12ms	600k
ListFlanT5-XLLTR	+10.68%	+5.69%	0.423	0.494	3s	3B
Ranking loss+Generative loss						
ListFlanT5-XLRankGen	-5.32%	-3.14%	0.341	0.406	12s	3B
Generative Loss						
FlanT5-XL-pointwise	-10.71%	-5.79%	0.303	0.379	2.1s	3B
FlanT5-XL-pairwise	-2.13%	-1.32%	0.371	0.432	22.34s	3B
FlanT5-XL-listwise	+3.25%	+1.79%	0.403	0.447	15.89s	3B
FlanT5-XL-setwise	+11.78%	+6.43%	0.539	0.519	10.13s	3B

CQ1: Ranking Performance: Table 1 reports ranking metric results for fine-tuned models on small NN networks and Flan-T5-XL network. We observe that ListLTR's performance is significantly better than DeepPLTR on MRR and NDCG metrics. This can be attributed to the interaction network and listwise loss, which better captures the interaction among queries mapped to a prefix and understands the graded relevance across all queries in the list. In contrast, the pairwise loss compares two queries at a time during training, and at inference, the score for each query in the list is computed independently. Both DeepPLTR and ListLTR models have inference latencies in the order of a few milliseconds, enabling real-time inference for production QAC models. Their high ranking performance is also attributed to high-quality customer behavior features collected and refreshed daily. The ListFlanT5-XLLTR model shows a marginal improvement over the ListLTR network on relevance. This model uses the Flan-T5-XL network's encoder for obtaining query embeddings and ListLTR as the score generator. The addition of embeddings from Flan-T5-XL provides a limited lift, as customer behavior (CB) features alone seem rich enough to bring maximum improvement in relevance. However, this network's efficiency is much poorer due to slow inference, making it unsuitable for deployment in a commercial real-time QAC setup.

As a next step, we try a combination of ranking and generative loss with the ListFlanT5-XLRankGen model, which uses the probability of generating a candidate as the score and employs listwise softmax loss with these scores in addition to generative loss for prefix-to-query generation. Note that in this sequence-to-

sequence setup, each prefix learns to generate a single query, and hence there is no interaction between queries at the time of scoring. The reported results are far poorer than the baseline, mainly because the queries are scored individually, and the knowledge imported from LLMs fails to outperform the CB features.
With Flan-T5-XL as the LLM ranking network and different prompting techniques to mimic the different LTR learning setups, we observe that pointwise and pairwise prompting techniques do not improve relevance over baselines. Moreover, with pairwise prompting, inference latency is huge as multiple calls are needed for sorting all the items in the list. Interestingly, listwise and setwise prompting techniques improve relevance over the baseline. However, FlanT5-XL-listwise relevance lift is unable to beat the lift with traditional ListLTR models. The FlanT5-XL-setwise model appears to be the best in terms of the lift in MRR and NDCG@10. The setwise prompting technique uses only the labels of the candidate queries at the time of training and inference, which simplifies the sequence-to-sequence task of reproducing the complete list of re-ranked candidates. Moreover, similar to listwise, it compares and contrasts a set of candidates together, making it efficient to capture listwise context. However, the inference latency for the setwise technique is in the order of a few seconds (far better than pairwise and listwise prompting techniques) but very poor compared to the baseline.

Table 2. Ranking and diversity performance of traditional and LLM-based ranking networks relative to DeepPLTR baseline. Relative lifts on e-commerce data; absolute numbers on public AOL dataset.

	E-commerce Dataset			AOL Dataset		
Model	MRR	NDCG@10	αNDCG@10	MRR	NDCG@10	αNDCG@10
Ranking loss						
DeepPLTR	-	-	-	0.384	0.438	0.578
moDPLTR	+2.48%	+1.03%	+2.24%	0.381	0.435	0.585
DiALTR	+9.23%	+5.16%	+22.83%	0.409	0.463	0.681
ListFlanT5-XLLTR	+10.47%	+5.33%	+28.91%	0.413	0.467	0.595
Generative Loss						
FlanT5-XL-setwise [COT]	+1.32%	+0.54%	+38.5%	0.386	0.459	0.713
FlanT5-XL-setwise [Instruct-FT]	+9.86%	+5.32%	+29.33%	0.406	0.455	0.692

CQ2: Diversity performance: Table 2 compares the performance of traditional models trained on diversity-aware ranking losses with Flan-T5-XL fine-tuned with prompts for diverse QAC ranking. While moDPLTR results in a marginal uplift in relevance and diversity metrics over the baseline, DiALTR significantly improves both metrics compared to the baseline. DiALTR uses ap-

proximate αNDCG as a loss function with listwise context during training, leading to better performance compared to the pairwise multi-objective fine-tuning in the case of moDPLTR. ListFlanT5-XLLTR, with Flan-T5-XL as the embedding generator and approximate αNDCG as the loss function, gives performance very similar to DiALTR, with no significant jump in the diversity metric, consistent with the trends observed in the relevance benchmarks reported in Table 2. FlanT5-XL-setwise [COT] results in a significant improvement in diversity metrics as it induces chain-of-thought prompting to first output a sequence of unique attributes and product phrases, followed by the construction of fluent queries. However, as this prompting forces the generation of sequences instead of labels, it leads to many new queries that may not be part of historical queries, resulting in a drop in ranking metrics. FlanT5-XL-setwise [Instruct-FT], on the other hand, instructs the model to re-rank the queries such that both relevance and diversity are balanced, with only re-ranked labels as output. We observe a marginal improvement in relevance and diversity metrics for this model compared to DiALTR. Although generative fine-tuning helps improve the diversity metric significantly, the improvement in the relevance metric is not enough to compensate for the latency and cost of inference.

CQ3: Latency vs performance discussion: Setwise prompting with FlanT5-XL demonstrates competitive performance compared to traditional listwise models on relevance and diversity metrics. However, the relevance lift is insignificant to replace traditional models due to huge latency. Distilling the LLM-based models, to improve latency while maintaining good relevance and diversity performance could lead to further relevance deterioration. Therefore, we leverage the diversity improvement of FlanT5-XL-setwise [COT] in an offline setting for QAC candidate generation, retaining the online performance of the traditional ranking model.

CQ4: Online A/B test: Utilizing the FlanT5-XL-setwise [COT] model, we generated diverse sequences incorporating attributes and categories for 100k prefixes. These candidate sequences were augmented with the existing pool of popular candidates and re-ranked for QAC using SOTA methods. The modified candidate generation setup underwent a month-long live A/B test on a large e-commerce store's search page, resulting in a 0.13% revenue lift and a 15% quality improvement in diversity and fluency compared to the baseline.

7.3 Conclusion

This study compares traditional models with Large Language Models (LLMs) on various parameters like model architecture, loss functions, and diversity objectives. Evaluating their generative capability and ranking potential through prompt/fine-tuning methods, we found LLMs superior in enhancing QAC diversity and relevance. However, they failed to meet real-time QAC latency requirements. To address this, we propose a practical approach combining language model capabilities with SOTA method efficiency, boosting revenue in a live A/B test. The study highlights LLMs' potential in QAC, paving the way for

e-commerce search to anticipate user needs and deliver a comprehensive shopping experience.

References

1. aol query log analysis. https://github.com/wasiahmad/aol_query_log_analysis/ (2017)
2. sentence transformers. https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1 (2022)
3. Bar-Yossef, Z., Kraus, N.: Context-sensitive query auto-completion. In: Proceedings of the 20th International Conference on World Wide Web (WWW). pp. 107–116 (2011)
4. Bhatia, S., Majumdar, D., Mitra, P.: Query suggestions in the absence of query logs. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. pp. 795–804 (2011)
5. Bruch, S., Wang, X., Bendersky, M., Najork, M.: An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In: Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval. pp. 75–78 (2019)
6. Bruch, S., Zoghi, M., Bendersky, M., Najork, M.: Revisiting approximate metric optimization in the age of deep neural networks. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval. pp. 1241–1244 (2019)
7. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
8. Cai, F., de Rijke, M.: Learning from homologous queries and semantically related terms for query auto completion. Information Processing & Management **52**(4), 628–643 (2016)
9. Cai, F., de Rijke, M.: A survey of query auto completion in information retrieval. Foundations and Trends® in Information Retrieval **10**(4), 273–363 (2016)
10. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th international conference on Machine learning. pp. 129–136 (2007)
11. Carbonell, J., Goldstein, J.: The use of mmr, diversity-based reranking for reordering documents and producing summaries pp. 335–336 (1998)
12. Chen, W., Liu, T.Y., Lan, Y., Ma, Z.M., Li, H.: Ranking measures and loss functions in learning to rank. Advances in Neural Information Processing Systems **22** (2009)
13. Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.M., Chen, W., et al.: Parameter-efficient fine-tuning of large-scale pre-trained language models. Nature Machine Intelligence **5**(3), 220–235 (2023)
14. Hou, Y., Zhang, J., Lin, Z., Lu, H., Xie, R., McAuley, J., Zhao, W.X.: Large language models are zero-shot rankers for recommender systems. In: Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part II. p. 364–381 (2024)
15. Jiang, J.Y., Ke, Y.Y., Chien, P.Y., Cheng, P.J.: Learning user reformulation behavior for query auto-completion. In: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval. pp. 445–454 (2014)

16. Kim, G.: Subword language model for query auto-completion. 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). pp. 5022–5032. Association for Computational Linguistics (2019)
17. Kulesza, A., Taskar, B.: Determinantal Point Processes for Machine Learning. Now Publishers Inc., Hanover, MA, USA (2012)
18. Nogueira, R., Jiang, Z., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. In: Findings of the Association for Computational Linguistics: EMNLP 2020 (2020)
19. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: Proceedings of the 1st international conference on Scalable information systems. pp. 1–es (2006)
20. Pasumarthi, R.K., Zhuang, H., Wang, X., Bendersky, M., Najork, M.: Permutation equivariant document interaction network for neural learning to rank. In: Proceedings of the 2020 ACM SIGIR on international conference on theory of information retrieval. pp. 145–148 (2020)
21. Pradeep, R., Sharifymoghaddam, S., Lin, J.: Rankvicuna: Zero-shot listwise document reranking with open-source large language models. arXiv preprint arXiv:2309.15088 (2023)
22. Qin, Z., Jagerman, R., Hui, K., Zhuang, H., Wu, J., Shen, J., Liu, T., Liu, J., Metzler, D., Wang, X., et al.: Large language models are effective text rankers with pairwise ranking prompting. In: Findings of the Association for Computational Linguistics: NAACL 2024 (2024)
23. Qin, Z., Yan, L., Zhuang, H., Tay, Y., Pasumarthi, R.K., Wang, X., Bendersky, M., Najork, M.: Are neural rankers still outperformed by gradient boosted decision trees? (2021)
24. Singh, S., Farfade, S., Comar, P.M.: Multi-objective ranking to boost navigational suggestions in ecommerce autocomplete. In: Companion Proceedings of the ACM Web Conference 2023. p. 469–474. WWW '23 Companion (2023)
25. Sun, W., Yan, L., Ma, X., Ren, P., Yin, D., Ren, Z.: Is chatgpt good at search? investigating large language models as re-ranking agent. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (2023)
26. Szpektor, I., Gionis, A., Maarek, Y.: Improving recommendation for long-tail queries via templates. In: Proceedings of the 20th international conference on World wide web. pp. 47–56 (2011)
27. Valizadegan, H., Jin, R., Zhang, R., Mao, J.: Learning to rank by optimizing ndcg measure. Advances in neural information processing systems **22** (2009)
28. Wang, S., Guo, W., Gao, H., Long, B.: Efficient neural query auto completion. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 2797–2804 (2020)
29. Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., Le, Q.V.: Finetuned language models are zero-shot learners. In: International Conference on Learning Representations (2022)
30. Xia, F., Liu, T.Y., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank: theory and algorithm. In: Proceedings of the 25th international conference on Machine learning. pp. 1192–1199 (2008)
31. Yan, L., Qin, Z., Pasumarthi, R.K., Wang, X., Bendersky, M.: Diversification-aware learning to rank using distributed representation. In: Proceedings of the Web Conference 2021. pp. 127–136 (2021)
32. Yuan, K., Kuang, D.: Deep pairwise learning to rank for search autocomplete. arXiv preprint arXiv:2108.04976 (2021)

33. Zhuang, H., Qin, Z., Jagerman, R., Hui, K., Ma, J., Lu, J., Ni, J., Wang, X., Bendersky, M.: Rankt5: Fine-tuning t5 for text ranking with ranking losses. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2308–2313 (2023)
34. Zhuang, S., Zhuang, H., Koopman, B., Zuccon, G.: A setwise approach for effective and highly efficient zero-shot ranking with large language models. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 38–47 (2024)