# Generative Product Recommendations for Implicit Superlative Queries

**Kaustubh D. Dhole**[α*], **Nikhita Vedula**[β], **Saar Kuzi**[β], **Giuseppe Castellucci**[β]
**Eugene Agichtein**[α*], **Shervin Malmasi**[β]
[α]Emory University, Atlanta, GA [β]Amazon.com Inc., Seattle, WA, USA
kdhole@emory.edu, {veduln,skuzi,giusecas,eugeneag,malmasi}@amazon.com

## Abstract

In recommender systems, users often seek the *best* products through indirect, vague, or under-specified queries, such as "*best shoes for trail running*". Such queries, also referred to as implicit superlative queries, pose a significant challenge for standard retrieval and ranking systems as they lack an explicit mention of attributes and require identifying and reasoning over complex attributes. We investigate how Large Language Models (LLMs) can generate implicit attributes for ranking as well as reason over them to improve product recommendations for such queries. As a first step, we propose a novel four-point schema for annotating the best product candidates for superlative queries called **SUPERB**, paired with LLM-based product annotations. We then empirically evaluate several existing retrieval and ranking approaches on our new dataset, providing insights and discussing their integration into real-world e-commerce production systems.

## 1 Introduction

Superlative queries are common in product search as users seek products with the highest degree of one or more attributes to satisfy their needs. While some superlative queries can be handled by existing retrieval systems (Kumar et al., 2024; Zhang et al., 2015) through attribute-based filtering (e.g., "the largest M2 Pro with 32 GB RAM"), others can pose challenges to the existing solutions.

Specifically, in this paper, we study the problem of product ranking and recommendation for *implicit superlative queries*, where the desired product attributes are not explicitly stated. These queries often involve aspects that require common sense knowledge of the product (Bos and Nissim, 2006; Scheible, 2007). This problem is further compounded by users creating vague and under specified search queries, either due to a lack of knowledge about certain entity features or the search spanning implicit dimensions, frequently leading to query-product mismatches. For example, a query such as "***the best toy for a 3 year old girl***" requires gauging the best products across several implicit attributes. To effectively serve such a query, product recommendations should consider popular toy standards like ASTM F963, quality, non-toxic materials, and bright, engaging colors — attributes that are often unknown to end users. With a plethora of product options available on e-commerce platforms, identifying the best products to meet customer needs requires additional product category and world knowledge.

| Queries | Query Type | Ranking Criteria |
|---|---|---|
| *toys* | Expecting Relevant Products | No Superlative criteria. |
| *highest rated toy for 3-year olds* | Objective Superlative | *Single Objective Criteria:* highest rating |
| *best toy for my 3-year nephew who loves the Flintstones* | Implicit Superlative | *Multiple & Implicit Criteria:* highly-rated, overall positively-reviewed, suitable for a male child, likes Flintstones, dinosaurs, etc. |

Table 1: Types of Queries along with the criteria of each. **SUPERB** focuses on implicit superlative queries.

Existing ranking pipelines (Reddy et al., 2022) rely on traditional relevance labels like 'Highly Relevant' vs 'Irrelevant' or ESCI (Exact, Substitute, Complement, Irrelevant), and are typically designed for highly objective queries. They do not capture the nuances of product quality and the subjective expectations of "best" products for a given need. In such a scenario, Large Language Models (LLMs) trained on vast amounts of data from diverse sources can act as sources of common-sense knowledge. They have been exposed to extensive text sources and have demonstrated success in modeling global opinions in various domains (Santurkar et al., 2023) and predicting user preferences (Kang et al., 2023). LLMs can leverage this knowledge to offer expert insights beyond the basic product descriptions, thereby enabling search and ranking based on external knowledge.

*Work done at Amazon.



*Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 77–91
April 30 - May 1, 2025 

We hypothesize that LLMs possess the capability to perform multi-objective optimization over implicit attributes that match user preferences. Hence, LLMs could play a pivotal role in recommending products for superlative queries by (i) offering comprehensive knowledge across multiple product dimensions and (ii) addressing the inherent subjectivity associated with such queries.

Our work aims to investigate the research question: ***Can LLMs effectively rank and recommend the "best" products?*** To that end, we propose a four-level labeling scheme for superlative queries – **SUPERB** with LLM-based annotations, and evaluate retrieval effectiveness across multiple traditional and LLM-based ranking pipelines. To our knowledge, this is the first work to explore implicit superlative queries for product recommendation. Specifically, we make the following contributions:

- We investigate the challenges in answering superlative queries, and define a four-level labeling scheme for relevance ratings.
- We introduce **SUPERB**,[1] **Super**latives with **B**est relevance annotations, a schema of superlative queries and pair them with LLM-based annotations using four different ranking approaches i.e., **pointwise**, **pairwise**, **listwise** and **deliberated** prompting.
- We evaluate the retrieval effectiveness of multiple ranking pipelines against **SUPERB**.

Our contributions highlight the importance of addressing superlative queries in recommendation systems, an area that has been largely overlooked.

## 2 Related Work

We now discuss related work to place our contributions in context.

### 2.1 LLMs for Ranking and Recommendation

LLMs have been successfully applied for ranking and recommendation (Yue et al., 2023). Early pointwise ranking approaches (Nogueira et al., 2019) fine-tuned BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020) with query-document pairs, and showed improved performance across a variety of benchmarks (Craswell et al., 2021; Thakur et al.). Pointwise approaches (Ma et al., 2024) ranked items based on scores predicted for individual documents, while pairwise approaches (Qin et al., 2024) prompted models with the query and two documents to compare and rank. Others (Pradeep et al., 2023a,b; Sun et al., 2023) explored a listwise ranking strategy by prompting with a list of documents and generating a ranked list of document IDs.

### 2.2 LLMs for Relevance Labelling

After showing promise in predicting searcher preferences (Thomas et al., 2024), LLMs have been extensively used in generating relevance labels (Faggioli et al., 2023; Yan et al., 2024; MacAvaney and Soldaini, 2023; Mehrdad et al., 2024; Dhole and Agichtein, 2024a; Dhole et al., 2025). As compared to human evaluation, automated relevance labeling is faster and more scalable.

### 2.3 Prompting Approaches

Apart from standard prompting approaches, deliberative prompting (Li et al., 2023; Zheng et al., 2024) approaches like Chain-of-Thought (Wei et al., 2022) and scaling inference time compute (Snell et al., 2024; Guo et al., 2025) have successfully improved the performance of LLMs. These methods involve the model generating related information, such as reasoning chains or explanations, to elucidate the reasoning process before arriving at an answer. Our deliberated prompting approach, discussed in Section 5.2 is on similar lines, where we seek to regurgitate implicit attributes so as to make them explicit and help in arriving at the appropriate best label.

### 2.4 Superlative Search Queries

Much of the research related to superlatives has focused on applications in question answering, opinion mining, and sentiment analysis. A recent study (Kumar et al., 2024) focused on ranking over objective superlatives where the dimensions to compare against (also referred to as the comparison set (Pyatkin et al., 2024)) are often explicitly provided. However, superlative queries often have implicit, vague and complex dimensions.

## 3 Implicit Superlative Queries

We now formalize the type of superlative queries that we seek to address. We define implicit superlative queries as those which (i) *seek the highest degree of one or more attributes or features of a product*; and (ii) *are implicit in nature*. These queries involve preferences which are generally popular, subjective, and not just based on quantifiable attributes. E.g., the superlative query "best toy

[1] https://github.com/emory-irlab/SUPERB

for my 3-year nephew who loves the Flintstones" – requires an implicit understanding that the user might be looking for a good quality toy which is well-rated and reviewed, reasonably priced, age appropriate, and relates to characters or properties of the show "The Flintstones". Addressing such implicit superlative queries would require (i) inferring hidden attributes, (ii) world knowledge or a general understanding of concepts, and (iii) being able to reason and compare across different related products and ensure that the necessary attributes are of the highest degree. Table 1 shows a summary and examples of targeted queries.

## 4 The SUPERB Relevance Scheme

We design a novel four-category relevance taxonomy to rank, recommend, and evaluate the retrieved product candidates for superlative queries.

- ***Overall Best (3)***: reserved for products that excel across a broad spectrum of parameters including quality, user experience, value for money, innovation, aesthetics, and environmental impact, among others. Products in this category represent the best of what is available in the market, meeting or exceeding all the expected criteria.
- ***Almost Best (2)***: includes products that perform exceptionally well for most criteria but may fall short in one or a few aspects. These products are generally considered top-tier but lack one or more elements that would elevate them to the Overall Best status.
- ***Relevant but Not the Best (1)***: captures products that are suitable for certain contexts or specific needs but do not represent the best available option across the board.
- ***Not Relevant (0)***: products that do not align well with the user's query or fail to meet the basic standards expected in their category, making them generally not recommended.

We design such a fine-grained system for multiple reasons. Fine-grained labels have been found to be more advantageous than simplistic binary choices (Zhuang et al., 2024). In addition, they facilitate nuanced evaluations and provide comprehensive feedback. For example, differentiating between **Overall Best** and **Almost Best** might be less obvious when purchasing standard office supplies, where basic functionality is adequate. However, this distinction becomes essential when selecting infant car seats, where the highest safety and technology standards are vital.

## 5 Dataset Construction

We now describe how we generate superlative queries and pair them with products labeled with annotations from our schema.

### 5.1 Creation of Superlative Queries

For generating superlative queries, we employ the Amazon Shopping Queries dataset (Reddy et al., 2022), which consists of search queries each annotated with up to 40 potential items with ESCI relevance judgements.[2]

Inspired by LLM-based reformulation approaches (Yang et al., 2023; Dhole and Agichtein, 2024b; Dhole et al., 2024), we prompt `Claude-Sonnet` (Anthropic, 2024b) with tailored few-shot instructions, to reformulate these shopping queries into their superlative counterparts. We select queries paired with at least five products with the **Exact** ESCI label. We consider all the products of such queries for subsequent **SUPERB** annotations.[3] We generate a total of 35,651 superlative queries from 1,825 original queries. The complete prompt is shown in Appendix Table 6 and some of the generated queries are shown in Table 2.

| Query | Superlative Queries |
|---|---|
| *"running shoes"* | *"best running shoes for flat feet"*<br>*"best running shoes for rocky terrain"* |
| *"diaper backpack"* | *"best diaper backpack for twins", "most comfortable diaper backpack for back pain"* |

Table 2: Examples of generated superlative queries.

### 5.2 Creating Relevance Annotations

We adopt four methods for annotating the retrieved product candidates with an LLM: **pointwise**, **pairwise**, **listwise** and **deliberated** prompting. In the **pointwise** approach, we prompt the model with a superlative query $q$ and the description of a product $p_1$, to generate a single annotation label $b_1$ that corresponds to a category in our schema, along with an explanation $E$ (Eq. 1).

$$(q, p_1) \rightarrow \mathbf{M} \rightarrow b_1 + E \quad (1)$$

[2]Exact (3), Substitute (2), Complement (1), Irrelevant (0)

[3]Products of the highest relevance might not necessarily be the **Overall Best** option.


Figure 1: Attributes generated through deliberated prompting for a superlative query.

In the **pairwise** approach, we want the model **M** to compare a product $p_1$ to another product $p_2$. Hence, we prompt **M** with the additional description $p_2$ and force it to generate two labels $b_1$ and $b_2$ for both products as shown (Eq. 2).

$$(q, p_1, p_2) \rightarrow \mathbf{M} \rightarrow b_1\, b_2 + E \quad (2)$$

In the **listwise** approach, we expand the context to $N-1$ additional products. We hypothesize that providing a context of other products would help the model make accurate judgements in inferring the necessary attributes. Besides, it is more efficient as compared to the pointwise approach as it can process multiple products simultaneously and generate category labels for each (Eq. 3).

$$(q, p_1, \ldots, p_N) \rightarrow \mathbf{M} \rightarrow b_1\, b_2\, \ldots\, b_N + E \quad (3)$$

The pairwise and listwise approaches allow gauging the properties of other related product(s) for generating the category label of a product. We do not explicitly force the model to select the highest category (i.e., Overall Best) in these scenarios.

$$q \rightarrow \mathbf{M} \rightarrow a_q \quad (4)$$

$$(q, a_q, p_1) \rightarrow \mathbf{M} \rightarrow b_1 + E \quad (5)$$

We also employ a two-step **deliberated prompting** strategy inspired by previous studies (Wei et al., 2022; Li et al., 2023; Zheng et al., 2024), which asks the model to deliberate and reason before generating the final answer. We first generate a set of attributes $a_q$ characterizing the best features of products, and then use them to prompt the model to generate the final taxonomy label (Eq. 4-5). These attributes serve as potential dimensions for the model to compare against in the subsequent pointwise step. Figure 2 shows an example of the label generation process with deliberated prompting.

In each of the methods, we also force the model to generate an explanation to improve model performance (Wei et al., 2022) and also aid human evaluation. Figure 1 shows sample generated attributes for a superlative query. We describe the corresponding instructions in Table 13 in the Appendix.

| Queries | Best Annotations |
|---|---|
| 2,230 | 29,218 |
| **Best Label** | **Number of Examples** |
| Overall Best | 8,564 |
| Almost Best | 10,100 |
| Relevant But Not the Best | 8,342 |
| Not Relevant | 2,212 |

Table 3: Category label distribution of **SUPERB**.

We use deliberated prompting to generate a large number of `(query, product, best-label)` triplets, which we refer to as **SUPERB**. We generate a total of 29,218 triplets corresponding to 2,230 randomly sampled unique superlative queries. The label distribution is shown in Table 3. Most of the labels are concentrated in the **Almost Best** and **Relevant But Not the Best** categories, with fewer in the **Not Relevant** category. This is expected as annotations were performed over products that were human-rated as **Exact**, albeit with respect to the original non-superlative queries.

## 6 Methods

We perform our analysis in a constrained setting where the item description is limited to 512 tokens in length. This is useful for low latency applications. We then use **SUPERB** for evaluating the following ranking pipelines:

(i) **BM25**: We use BM25 as our baseline.

(ii) **RM3**: We also employ a pseudo-relevance feedback baseline RM3 (Abdul-Jaleel et al., 2004).

(iii) **BM25/RM3 + Listwise Re-ranking**: Here, we re-rank the results of the first stage BM25 and RM3 through a listwise ranking approach. We force the model to generate a ranked list of product IDs in the style of RankGPT (Sun et al., 2023) (Eq. 6).

$$(q, p_1, \ldots, p_N) \rightarrow \mathbf{M} \rightarrow r_1\, \ldots\, r_N + E \quad (6)$$

where $r_j$ is the index of a product ranked $j$.

(iv) **BM25/RM3 + Deliberated Pointwise Re-ranking**: Here, the model is forced to generate a schema label for each item along with a confidence score, when given a query and estimated product attributes. The final ranked list is obtained by first sorting using the labels, and resolving ties first by confidence scores, and then by the BM25 scores.

| Retrieval Pipeline | P@5 | P@10 | P@20 | nDCG@5 | nDCG@10 | nDCG@20 |
|---|---|---|---|---|---|---|
| BM25 | .206 | .163 | .125 | .219 | .213 | .235 |
| RM3 | .214 | .180 | .139 | .219 | .219 | .243 |
| BM25 Top K + Pointwise Reranking | .226 | .163 | - | .205 | .198 | - |
| RM3 Top K + Pointwise Reranking | .208 | .180 | - | .199 | .210 | - |
| BM25 Top K + Listwise Reranking | **.262**$^{\alpha}$ | .192$^{\alpha}$ | .125 | **.278**$^{\alpha}$ | **.259**$^{\alpha}$ | **.264**$^{\alpha}$ |
| RM3 Top K + Listwise Reranking | .248 | **.201**$^{\alpha}$ | **.140** | .245 | .241 | .254 |

Table 4: Performance metrics for different ranking pipelines. $\alpha$ denotes significant improvements (paired t-test with Holm-Bonferroni (Holm, 1979) correction, p < 0.05) over BM25.

| Retrieval Pipeline | P@10 | P@50 | nDCG@10 | nDCG@50 |
|---|---|---|---|---|
| BM25-Top 100 | .154 | .079 | .205 | .279 |
| BM25-Top 100 + Window (5, 2) | **.185**$^{\alpha}$ | **.084**$^{\alpha}$ | **.241**$^{\alpha}$ | **.309**$^{\alpha}$ |
| BM25-Top 100 + Window (20, 10) | .198$^{\alpha}$ | .082$^{\alpha}$ | .240 | .302$^{\alpha}$ |
| BM25-Top 200 | .196 | .079 | .205 | .279 |
| BM25-Top 200 + Window (20, 10) | **.262** | **.088** | **.259** | **.328** |

Table 5: Comparing different retrieval pipelines for the long context setting. $\alpha$ denotes significant improvements (paired t-test with Holm-Bonferroni (Holm, 1979) correction, p < 0.05) over BM25.

This can also be seen as a black-box counterpart of pointwise ranking approaches which provide confidence through logit probabilities. The confidence scores range between 1 and 9 (Eq. 7-8).

$$q \rightarrow \mathbf{M} \rightarrow a_q \tag{7}$$

$$(q, p_1, a_q) \rightarrow \mathbf{M} \rightarrow b_1 + c_1 + E \tag{8}$$

We choose the `Claude-Haiku` (Anthropic, 2024a) model for our experiments since it is beneficial to evaluate smaller models for production pipelines. We use the PyTerrier (Macdonald and Tonellotto, 2020) library with the PyTerrier-GenRank (Dhole, 2024) plugin for designing the retrieval and re-ranking pipelines, and computing `precision` and `nDCG` metrics.

**Analysis on Longer Context**: We also analyze the case where we use **longer product descriptions**, and when there are a **large number of products in the context**. In that case, employing a listwise strategy can be detrimental as LLMs have been known to show bias towards specific positions of text in the context (Liu et al., 2024), while employing a pointwise strategy would involve excessive inference calls. Also, in practice, we found that LLMs find it hard to generate 100 or 200 item IDs at once hindering their ability to rerank items properly. We hence evaluate such queries using (v) a **BM25 + Sliding-window** approach introduced in RankGPT (Sun et al., 2023).

## 7 Results and Analysis

As shown in Table 4, we find that the listwise ranking approach is able to rank the best products significantly better as compared to other approaches across all metrics. The listwise scores are better for queries with larger `nDCG` values of BM25 meaning they benefit from an initial ranked list as shown in Appendix Figure 4. Pointwise approaches also help marginally with `P@10` compared to BM25.

We also show the results for top-100 and top-200 items with long descriptions in Table 5. We find that employing a listwise approach in a sliding window fashion significantly improves retrieval effectiveness over the baseline BM25 retrieval across all metrics. In some cases, we observed modest improvements compared to BM25, highlighting the difficulty of handling superlative queries, which is inherently challenging due to ambiguities and the need for extensive world knowledge. This complexity underscored the hardness of the task, as it requires more than traditional retrieval models.

### 7.1 Error Analysis

By analyzing queries where the methods perform well or poorly, we can gain insights into the model's behavior. The relative performance by nDCG@10 is summarized in Figure 4 in the Appendix.

**Both BM25 and LLM perform well:** Queries like "most versatile baby carrier for all terrains"

(nDCG@10 of 0.756 and 0.756, respectively) and "Best of montreal album for summer road trips" (0.787, 0.951) show strong performance for both approaches. These queries are specific, and the attributes are commonly matched both lexically and semantically to product descriptions.

**Both BM25 and LLM perform poorly:** For queries such as "Most durable kids plates not plastic" (nDCG@10 of 0.024 and 0.016, respectively) and "most gentle water wipes for baby's skin" (nDCG 0.066 and 0.054, respectively), both approaches struggled. In these cases, challenges like negation, tokenization errors, and specific attributes may contribute to poor performance.

**LLM outperforms BM25:** Queries like "most modern LG refrigerators to complement minimalist kitchen decor" or "most stylish child safety harness to match toddler's outfits" involve interpreting nuances related to style, versatility, and aesthetics, where LLMs arguably excel i.e. recognizing global preferences and broader contexts, enabling them to rerank products with less tangible attributes.

**BM25 outperforms LLM:** Many of the BM25-favored queries have clear, well-defined criteria, such as "safest bottle warmer for preserving nutrients" (nDCG 0.508 vs. 0.264); "most flexible rv caulking sealant for easy application" (nDCG 0.619 vs. 0.474). We speculate that BM25 excels with queries containing specific product terms and common words, as it performs well without advanced reasoning, while LLMs might over-generalize.

## 8 Conclusion

This work studied superlative queries with implicit attributes, which are typically more complex compared to other query types since ranking products for them requires inferring attributes, placing other products in context, and using commonsense knowledge to determine the best ones. Our analysis shows that LLMs can rank the best items, improve ranking when provided with initial ranked lists, and can also be sensitive to them. In addition, our methods are applicable to rank superlative queries in other item and document ranking settings.

We present the **SUPERB**, **4-point schema** and propose **pointwise, deliberated pointwise, pairwise, and listwise** methods to label superlative queries over it and re-rank retrieved products, using an LLM as the backbone. The listwise approach is preferable for lower budgets, while the deliberated point-wise approach can be preferred for better quality annotations. We believe that our study can drive further research on superlative search queries.

Our work highlights key considerations for deploying an LLM-based product ranking system into production. While a listwise approach effectively ranks multiple items at once, it can be inefficient due to lengthy item descriptions. In contrast, a pointwise approach is faster, especially with parallel processing. Sliding window methods and query reformulation are also viable alternatives. Generating attributes and explanations clarifies label assignments, boosting user trust and satisfaction.

Addressing superlative queries in product recommendation systems is essential, particularly for the next generation of interactive shopping assistants (Vedula et al., 2024; Li et al., 2025) and generative recommender systems (Senel et al., 2024). This becomes even more relevant as information-seeking and product search system grow closer together (Kuzi and Malmasi, 2024). These superlative queries capture user intent to find the best possible items, an aspect often overlooked in current systems. Introducing **SUPERB** allows for the development and assessment of recommendation pipelines capable of handling high-expectation queries, helping systems address this unmet need.

## Limitations

LLMs have a tendency to average out preferences and often aligning to the majority of the users making them apt for our use case, as shoppers frequently tend to buy the best products unanimously for instance, following viral trends or popular recommendations provided by bloggers.

However, there are other types of superlative queries that could be subjective and depend on user preferences. It would be interesting to see how such user preferences could be incorporated in ranking the best. We envisage various ways our work could be extended to achieve this – through traditional techniques like relevance feedback, conversational interactions, and understanding cultural contexts (Dhole, 2023; Mitchell et al., 2025). Besides, users often make use of public reviews, blogs and ephemeral trends to guide their purchase decision (Hsu et al., 2013; Wilson et al., 2024). Hence incorporating public reviews, and external information through retrieval augmentation could be an interesting line of subsequent study.

## Acknowledgments

The authors would like to thank Dhineshkumar Ramasubbu for helping with the annotations and the anonymous reviewers for their helpful feedback.

## References

Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard. *Computer Science Department Faculty Publication Series*, page 189.

Anthropic. 2024a. Claude 3 haiku: our fastest model yet. Accessed: 2024-07-10.

Anthropic. 2024b. Introducing claude 3.5 sonnet. Accessed: 2024-07-10.

Johan Bos and Malvina Nissim. 2006. An empirical approach to the interpretation of superlatives. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 9–17.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. Ms marco: Benchmarking ranking models in the large-data regime. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 1566–1576.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Kaustubh Dhole. 2023. Large language models as sociotechnical systems. In *Proceedings of the Big Picture Workshop*, pages 66–79.

Kaustubh Dhole and Eugene Agichtein. 2024a. Llm judges for retrieval augmented argumentation.

Kaustubh D Dhole. 2024. Pyterrier-genrank: The pyterrier plugin for reranking with large language models.

Kaustubh D Dhole and Eugene Agichtein. 2024b. Genqrensemble: Zero-shot llm ensemble prompting for generative query reformulation. In *European Conference on Information Retrieval*, pages 326–335. Springer.

Kaustubh D Dhole, Ramraj Chandradevan, and Eugene Agichtein. 2024. Generative query reformulation using ensemble prompting, document fusion, and relevance feedback. *arXiv preprint arXiv:2405.17658*.

Kaustubh D. Dhole, Kai Shu, and Eugene Agichtein. 2025. ConQRet: Benchmarking fine-grained evaluation of retrieval augmented argumentation with LLM judges. In *Proceedings of the 2025 Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, Albuquerque, New Mexico. Association for Computational Linguistics.

Guglielmo Faggioli, Laura Dietz, Charles LA Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, et al. 2023. Perspectives on large language models for relevance judgment. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 39–50.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

Chin-Lung Hsu, Judy Chuan-Chuan Lin, and Hsiu-Sen Chiang. 2013. The effects of blogger recommendations on customers' online shopping intentions. *Internet research*, 23(1):69–88.

Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. 2023. Do llms understand user preferences? evaluating llms on user rating prediction. *arXiv preprint arXiv:2305.06474*.

Nitesh Kumar, Usashi Chatterjee, and Steven Schockaert. 2024. Ranking entities along conceptual space dimensions with llms: An analysis of fine-tuning strategies. *arXiv preprint arXiv:2402.15337*.

Saar Kuzi and Shervin Malmasi. 2024. Bridging the Gap Between Information Seeking and Product Search Systems: Q&A Recommendation for E-Commerce. *SIGIR Forum*, 58(1):1–10.

Bei Li, Rui Wang, Junliang Guo, Kaitao Song, Xu Tan, Hany Hassan, Arul Menezes, Tong Xiao, Jiang Bian, and JingBo Zhu. 2023. Deliberate then generate: Enhanced prompting framework for text generation. *arXiv preprint arXiv:2305.19835*.

Xiangci Li, Zhiyu Chen, Jason Ingyu Choi, Nikhita Vedula, Besnik Fetahu, Oleg Rokhlenko, and Shervin Malmasi. 2025. Wizard of shopping: Target-oriented e-commerce dialogue generation with decision tree branching.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA. Association for Computing Machinery.

Sean MacAvaney and Luca Soldaini. 2023. One-shot labeling for automatic relevance estimation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2230–2235, New York, NY, USA. Association for Computing Machinery.

Craig Macdonald and Nicola Tonellotto. 2020. Declarative experimentation ininformation retrieval using pyterrier. In *Proceedings of ICTIR 2020*.

Navid Mehrdad, Hrushikesh Mohapatra, Mossaab Bagdouri, Prijith Chandran, Alessandro Magnani, Xunfan Cai, Ajit Puthenputhussery, Sachin Yadav, Tony Lee, ChengXiang Zhai, et al. 2024. Large language models for relevance judgment in product search. *arXiv preprint arXiv:2406.00247*.

Margaret Mitchell, Giuseppe Attanasio, Ioana Baldini, Miruna Clinciu, Jordan Clive, Pieter Delobelle, Manan Dey, Sil Hamilton, Timm Dill, Jad Doughman, Ritam Dutt, Avijit Ghosh, Jessica Zosa Forde, Carolin Holtermann, Lucie-Aimée Kaffee, Tanmay Laud, Anne Lauscher, Roberto L Lopez-Davila, Maraim Masoud, Nikita Nangia, Anaelia Ovalle, Giada Pistilli, Dragomir Radev, Beatrice Savoldi, Vipul Raheja, Jeremy Qin, Esther Ploeger, Arjun Subramonian, Kaustubh Dhole, Kaiser Sun, Amirbek Djanibekov, Jonibek Mansurov, Kayo Yin, Emilio Villa Cueva, Sagnik Mukherjee, Jerry Huang, Xudong Shen, Jay Gala, Hamdan Al-Ali, Tair Djanibekov, Nurdaulet Mukhituly, Shangrui Nie, Shanya Sharma, Karolina Stanczak, Eliza Szczechla, Tiago Timponi Torrent, Deepak Tunuguntla, Marcelo Viridiano, Oskar van der Wal, Adina Yakefu, Aurélie Névéol, Mike Zhang, Sydney Zink, and Zeerak Talat. 2025. SHADES: Towards a multilingual assessment of stereotypes in large language models. In *Proceedings of the 2025 Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, Mexico City, Mexico. Association for Computational Linguistics.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Valentina Pyatkin, Bonnie Webber, Ido Dagan, and Reut Tsarfaty. 2024. Superlatives in context: Explicit and implicit domain restrictions for superlative frames. *arXiv preprint arXiv:2405.20967*.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, Mexico City, Mexico. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. 2022. Shopping queries dataset: A large-scale ESCI benchmark for improving product search.

Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. 2023. Whose opinions do language models reflect? In *International Conference on Machine Learning*, pages 29971–30004. PMLR.

Silke Scheible. 2007. Towards a computational treatment of superlatives. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 67–72, Prague, Czech Republic. Association for Computational Linguistics.

Lütfi Kerem Senel, Besnik Fetahu, Davis Yoshida, Zhiyu Chen, Giuseppe Castellucci, Nikhita Vedula, Jason Choi, and Shervin Malmasi. 2024. Generative Explore-Exploit: Training-free Optimization of Generative Recommender Systems using LLM Optimizers. In *Proceedings of ACL 2024 (Research Track)*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models.

Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2024. Large language models can accurately predict searcher preferences. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 1930–1940, New York, NY, USA. Association for Computing Machinery.

Nikhita Vedula, Oleg Rokhlenko, and Shervin Malmasi. 2024. Question suggestion for conversational shopping assistants using product metadata. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2960–2964, New York, NY, USA. Association for Computing Machinery.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

George Wilson, Oliver Johnson, and William Brown. 2024. The influence of digital marketing on consumer purchasing decisions.

Le Yan, Zhen Qin, Honglei Zhuang, Rolf Jagerman, Xuanhui Wang, Michael Bendersky, and Harrie Oosterhuis. 2024. Consolidating ranking and relevance predictions of large language models through post-processing. *arXiv preprint arXiv:2404.11791*.

Dayu Yang, Yue Zhang, and Hui Fang. 2023. Zero-shot query reformulation for conversational search. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 257–263.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. Llamarec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089*.

Sheng Zhang, Yansong Feng, Songfang Huang, Kun Xu, Zhe Han, and Dongyan Zhao. 2015. Semantic interpretation of superlative expressions via structured knowledge bases. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 225–230.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2024. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 358–370.

## Appendix

Given a query, generate multiple diverse superlative versions of the same which require common sense inference. The reformulated superlative queries should provide additional context for which common sense knowledge is required. The context should be related to the item in the original query in various ways and should seek the highest degree of some related aspects. For instance, if a user is looking for a mouse pad, she might be interested in the best one which best complements the color of her laptop, or may require the most suitable one for painful wrists, etc. The context should require generally understood knowledge and common sense and it should not depend on objective criteria like highest rated or cheapest. Some examples of superlative queries are “Best booster chairs to make mealtime hassle-free for my toddler”, “most user-friendly diaper pail to make my life as a new mom easier”, “most suitable lawnmover for rocky areas”, “most stylish and modern changing table pad to complement my nursery decor”,“Smoothest-riding 2 seater stroller for twin toddlers”,“Best diaper genie for sparking a child’s creativity”,“Highest quality epoxy resin for creating stunning wood art pieces”, You should not try to change the type of the product which the user is asking for. Only if the product explicitly mentions a single product, you should change it to make it more generalized (for instance, Amazon $100 gift card can be changed to $100 gift card and so on). Do not generate anything else except for one body of JSON and do not explain yourself. Do not include double quotes while generating the superlatives.
Provide your output in the form of a JSON.
Input Query: LEGO kit
{{
“superlatives” : [
“best LEGO kit for chess players”,
“best lego kits for marvel fans”,
“most impressive lego kits for my friend who is fascinated about India”,
“best lego kit to encourage my toddler to learn astronomy”,
]
}}

Input Query: black halter beaded satin long gowns sequin
{{
“superlatives”: [
“Trendiest black halter beaded satin long gowns with sequins for an Afro-themed fashion parade”,
“Best halter beaded satin long gowns to match my husband’s black silk coat”,
“Most casual black halter satin long gowns with sequins helpful ”,
“most suitable black halter beaded satin long gowns sequin for a date night”
]
}}

Input Query: armani exchange glasses
{{
“superlatives”: [
“best glasses with bold and trendy frames”,
“best glasses which can be used for office and at parties”,
“best retro look armani exchange glasses”,
“most suitable armani exchange glasses for travelling to dubai and mexico" ,
“best armani exchange glasses that blend seamlessly with my red jeans”,
]
}}

Input Query: {query}

Table 6: Prompt used for Superlative Query Generation

Based on the item description and some of its reviews, your internal knowledge about all the features of such types of items, and a user's given shopping query, you should classify the item into one of the taxonomy categories:

User Query: {query}
Item Description: Title: {title} Description: {description}
User Query: {query}

Categories:
3. Overall Best: The item meets the following criteria: The item is overall best in its category on various parameters – excellence in quality, user experience, value for money, innovation, aesthetics, environmental impact, market position, safety, versatility, processing speed, user rating, etc..
2. Almost Best: The item scores high on most or majority of the parameters except for a few. Most users would consider this as item as the best..
1. Relevant But Not Best: The item is suitable in certain contexts but not the best option..
0. Not Relevant: The item is generally not recommended as it is not relevant to the user's query..

Please classify the item into one of the four types. You should return a number between between 3 (Overall Best) and 0 (Not Relevant) followed by an explanation on the next line justifying why that category of best is suitable.

Table 7: Pointwise Prompt Used For Best Annotations

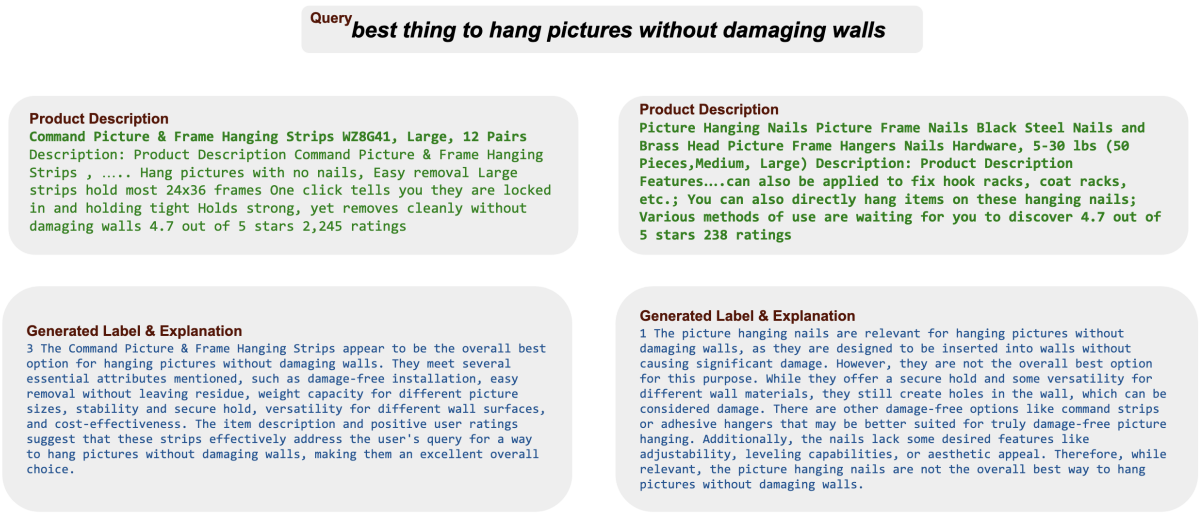


Figure 2: Sample generated label and explanation using the deliberated pointwise approach.

## A Evaluating the Best Product Judgements

To evaluate the efficacy of the **SUPERB** labels from the above methods, we perform a human evaluation to record the agreement with the model's labels. In-house domain experts performed the annotation. For each superlative query, the product descriptions, the corresponding category labels and their explanations from the pointwise, pairwise and listwise methods are presented to the annotator, who may agree with none, some, or all of the LLM generated labels.

As shown in Table 8, in our first phase of human evaluation, we find that the pointwise approach is more often preferred over listwise and pairwise approaches. During the process of annotation, we find that the pairwise approach tends to narrow its focus on attributes presented in the single product in the context, often misjudging necessary attributes. In the pointwise and listwise approaches, this seems to be less of a concern.

In the second phase of human evaluation, we use the best strategy of the first phase, i.e., pointwise, and measure the effects of deliberation over a separate set of queries. We find that deliberated prompting is preferred more often than its non-deliberated counterpart, as shown in Table 9, and making the attributes explicit helps assign better quality annotations.

| | Pointwise | Pairwise | Listwise |
|---|---|---|---|
| Agreement Rate | **66.36%** | 44.86% | 60.75% |

Table 8: Comparing the three best labelling approaches over 107 random superlative queries.

| | Without | With Deliberation |
|---|---|---|
| Agreement Rate | 75.23% | **78.90%** |

Table 9: Effect of deliberation on pointwise prompting for 109 random superlative queries.

**Effect of Increasing the Number of Products:** We measure the listwise ranking performance while increasing the number of input products $K$. As shown in Figure 3, we find that the listwise approach increases the likelihood of picking the best product as we provide more products in the context, and then tends to stagnate after a large $K$. The pointwise approach's performance remains almost the same.

As shown in Table 10, we also shuffle the product order from the first stage retriever and evaluate how sensitive the listwise re-ranker is to the initial order. Shuffling the top-20 products in three different random orders causes drastic performance drops in each, i.e. listwise re-ranking benefits from an initial ranked list and improves upon it.

| Listwise | BM25 | seed1 | seed2 | seed3 | RM3 |
|---|---|---|---|---|---|
| nDCG@10 | .259 | .147 | .143 | .141 | .241 |

Table 10: Listwise reranking performance when the top-20 products are placed in context with initial rankings from BM25, random and RM3 orderings. The listwise re-ranker is highly sensitive to the order provided by the first stage retriever.

## B Effect of Query Reformulation

To reduce inference latency for such scenarios, we also investigate incorporating LLM-based reformulation i.e. employing the LLM during query generation rather than during reranking. Specifically, we introduce (vi) two types of **query reformulations** to generate i) ***keywords***: this is accomplished by generating generic query expansion terms which are related to the query ii) ***attributes***: we use the above estimated ideal attributes for expanding the query.

**Results:** We also find that employing keyword and attribute-based reformulated queries helps improve overall retrieval effectiveness, as compared to the original queries. Attribute-based reformulation improves recall and MAP across all retrieval settings.

We find that by employing keyword and attribute based reformulated queries helps improve overall retrieval effectiveness, as compared to the original queries. Attribute based reformulation improves recall and MAP across all retrieval settings. Table 11 presents the details.

Based on the following descriptions of multiple items and a user's shopping query, you need to classify each item into one of the taxonomy categories:

User Query: {query}
Item 1 Description: Title: {Title 1} Description: {Item Description 1}
Item 2 Description: Title: {Title 2} Description: {Item Description 2}
...
...
Item N-1 Description: Title: {Title N-1} Description: {Item Description N-1}
Item N Description: Title: {Title N} Description: {Item Description N}
User Query: {query}

Classification Categories:

3. Overall Best: The item meets the following criteria: The item is overall best in its category on various parameters – excellence in quality, user experience, value for money, innovation, aesthetics, environmental impact, market position, safety, versatility, processing speed, has been rated highly, etc..
2. Almost Best: The item scores high on most or majority of the parameters except for a few. Most users would consider this as item as the best..
1. Relevant But Not Best: The item is suitable in certain contexts but not the best option..
0. Not Relevant: The item is generally not recommended as it is not relevant to the user's query..

Please rank each item into one of the four types. First, return the rankings as numbers separated by ' ' where each number ranges between between 3 (Overall Best) and 0 (Not Relevant). And then provide a short explanation as to why you assigned the best categories. You should start your answer with only the rankings (i.e. 3 2 2 0 and so on ) and not a description. Ensure that the number of rankings is equal to the number of items shown i.e. exactly 25.

Table 12: Listwise Prompt Used For Best Annotations – Provides multiple additional items as context

Given a user seeking the best item, define the ideal requirements for satisfying the user query by returning a list of attributes which are essential for that item. For instance, if the user is seeking the best laptop for his 15 year old son, the attributes could be a large RAM, the best GPUs (maybe from NVIDIA or AMD), good speakers etc. You should try to come up attributes which are essential for the perfect or the best item as well as which satisfy the user query. Return your output as a json. Do not generate anything else. {query}

Table 13: Deliberation Step used for Generating Attributes

Based on the following descriptions of two items, their reviews, and a user's shopping query, you need to rank each item into one of the taxonomy categories:

User Query: {query}
Item 1 Description: Title: {Title 1} Description: {Item Description 1}
Item 2 Description: Title: {Title 2} Description: {Item Description 2}
User Query: {query}

Categories:
3. Overall Best: The item meets the following criteria: The item is overall best in its category on various parameters – excellence in quality, user experience, value for money, innovation, aesthetics, environmental impact, market position, safety, versatility, processing speed, has been rated highly, etc..
2. Almost Best: The item scores high on most or majority of the parameters except for a few. Most users would consider this as item as the best..
1. Relevant But Not Best: The item is suitable in certain contexts but not the best option..
0. Not Relevant: The item is generally not recommended as it is not relevant to the user's query..

Please rank each item into one of the four types. First, return two numbers separated by ' ' where each number ranges between between 3 (Overall Best) and 0 (Not Relevant). And then briefly explain why the category of best is suitable.

Table 14: Pairwise Prompt Used For Best Annotations – Provides one additional item as context

Based on the following descriptions of two items, their reviews, and a user's shopping query, you need to rank each item into one of the taxonomy categories:
User Query: {query}
The best item would possibly possess many of such attributes: {Predicted Attributes}
Item 1 Description: Title: {title} Description: {Item Description}
User Query: {query}
Categories:
3. Overall Best: The item meets the following criteria: The item is overall best in its category on various parameters – excellence in quality, user experience, value for money, innovation, aesthetics, environmental impact, market position, safety, versatility, processing speed, has been rated highly, etc..
2. Almost Best: The item scores high on most or majority of the parameters except for a few. Most users would consider this as item as the best..

1. Relevant But Not Best: The item is suitable in certain contexts but not the best option..
0. Not Relevant: The item is generally not recommended as it is not relevant to the user's query..

Please rank each item into one of the four types. First, return two numbers separated by ' ' where each number ranges between between 3 (Overall Best) and 0 (Not Relevant). And then briefly explain why the category of best is suitable.

Table 15: Deliberated Pointwise Prompt Used For Best Annotations – Predicted attributes are provided as context

Based on the item description and some of its reviews, your internal knowledge about all the features of such types of items, and a user's given shopping query, you should classify the item into one of the taxonomy categories and provide a confidence score for your prediction:

User Query: {query}
The best item would possibly possess many of such attributes: {Predicted Attributes}
Item Description: Title: {title} Description: {description}
User Query: {query}

Categories:
3. Overall Best: The item meets the following criteria: The item is overall best in its category on various parameters – excellence in quality, user experience, value for money, innovation, aesthetics, environmental impact, market position, safety, versatility, processing speed, user rating, etc..
2. Almost Best: The item scores high on most or majority of the parameters except for a few. Most users would consider this as item as the best..
1. Relevant But Not Best: The item is suitable in certain contexts but not the best option..
0. Not Relevant: The item is generally not recommended as it is not relevant to the user's query..

You should return a number between between 3 (Overall Best) and 0 (Not Relevant) followed by the confidence of your prediction between 1 to 9 and an explanation on the next line justifying why that category of best is suitable. Your output should look something like this: 2 8 some explanation or 3 4 some explanation. If you are fully confident, then your confidence should have high values like 7, 8 upto 9. If you are not sure, then you should assign low confidence values like 1, 2 or 3. If you are partially confident, then assign other values.

Table 16: Deliberated Pointwise Prompt Used For Ranking for generating labels and confidence scores.

Based on the following descriptions of multiple items and a user's shopping query, you need to rank the items using the below taxonomy:

User Query: {query}
Item 1 Description: Title: {Title 1} Description: {Item Description 1}
Item 2 Description: Title: {Title 2} Description: {Item Description 2}
...
...
Item N-1 Description: Title: {Title N-1} Description: {Item Description N-1}
Item N Description: Title: {Title N} Description: {Item Description N}
User Query: {query}

Classification Categories:
3. Overall Best: The item meets the following criteria: The item is overall best in its category on various parameters – excellence in quality, user experience, value for money, innovation, aesthetics, environmental impact, market position, safety, versatility, processing speed, has been rated highly, etc..
2. Almost Best: The item scores high on most or majority of the parameters except for a few. Most users would consider this as item as the best..
1. Relevant But Not Best: The item is suitable in certain contexts but not the best option..
0. Not Relevant: The item is generally not recommended as it is not relevant to the user's query..

The 'Overall Best' item(s) should be ranked higher, followed by the 'Almost Best' item(s), the 'Relevant But not the best' and then the 'not relevant' ones. You should return the item ids separated by ' ' something like 8 3 9 1 2... You should start your answer with only the rankings and not a description. Ensure that each item id is present in the list. Ensure that the number of rankings is equal to the number of items shown i.e. exactly $K$.

Table 17: Listwise Prompt Used For Ranking

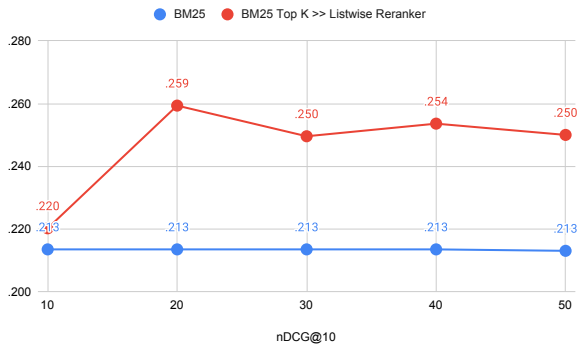


Figure 3: Listwise ranking consistently improves best ranking for different values of K.

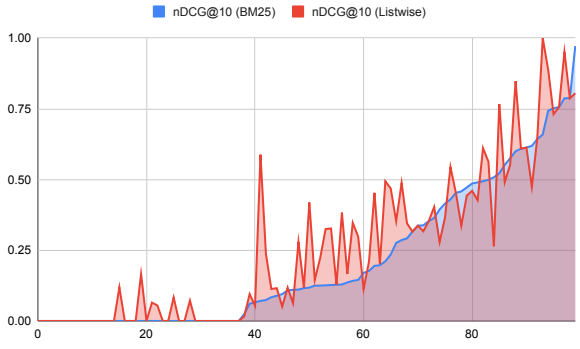


Figure 4: Listwise scores rank better than BM25 for almost all queries. Moreover, LLMs when employed in a listwise fashion benefit from an initial ranked list as queries with higher BM25 scores tend to get better improvements from the listwise approach.

| | BM25 | | | BM25 + Window (20,10) | | |
|---|---|---|---|---|---|---|
| Queries | MAP | R@50 | **nDCG@50** | MAP | R@50 | **nDCG@50** |
| SUPERB (Raw) | .152 | .358 | .279 | .168 | .372 | .302 |
| + Keyword based QR | .155 | .371 | **.291** | .172 | .383 | .31 |
| + Attribute based QR | **.156** | **.382** | **.291** | **.176** | **.389** | **.311** |

Table 11: Comparison of Query Reformulation with BM25 over superlative queries.