Leveraging LLMs to Enhance a Web-Scale Webpage Recommendation System

Iman Barjasteh*, Jaidev Shah*, Amey Barapatre*, Rana Forsati*, Gang Luo*
Fan Wu, Julie Fang, Xue Deng, Blake Shephard, Ronak Shah, Linjun Yang, Hongzhi Li, Rangan Majumder
{imbarjas,jaidevshah,abarapatre,raforsat,gluo}@microsoft.com
{fwu,juliefang,xuedeng,blakesh,rosh,linjya,hongzl,ranganm}@microsoft.com
Microsoft AI, USA

Abstract

Explore Further @ Bing is a webpage-to-webpage recommendation product, enhancing the search experience on Bing by surfacing engaging webpage recommendations tied to the search result URLs. In this paper, we present our approach for leveraging Large Language Models (LLMs) for enhancing our web-scale recommendation system. We describe the development and validation of our LLM-powered recommendation quality metric RecoDCG. We discuss our core techniques for utilizing LLMs to make our ranking stage quality-aware. Furthermore, we detail Q' recall, a recall path that enhances our system's candidate generation stage by leveraging LLMs to produce complementary and engaging recommendation candidates. We also address how we optimize our system for multiple objectives, balancing recommendation quality with click metrics. We deploy our work to production, achieving a significant improvement in recommendation quality. We share results from offline and online experiments as well as insights and steps we took to ensure our approaches scale effectively for our web-scale needs.

CCS Concepts

• **Information systems → Recommender systems**.

Keywords

Large Language Models, Quality, Ranking, Recall, Multitask, Large Scale Recommender System

ACM Reference Format:
Iman Barjasteh*, Jaidev Shah*, Amey Barapatre*, Rana Forsati*, Gang Luo and Fan Wu, Julie Fang, Xue Deng, Blake Shephard, Ronak Shah, Linjun Yang, Hongzhi Li, Rangan Majumder. 2024. Leveraging LLMs to Enhance a Web-Scale Webpage Recommendation System. In *Proceedings of EARL '24: Workshop on Evaluating and Applying Recommendation Systems with Large Language Models at RecSys '24.* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

*These authors contributed equally to this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
EARL '24: Workshop on Evaluating and Applying Recommendation Systems with Large Language Models at RecSys '24, October 14–18, 2024, Bari, Italy
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Explore Further, shown in Figure 1, lives as a feature on Bing search result page and offers query and user agnostic webpage recommendations. Our recommendation system is one of the largest in industry, with a total index size of over 200 billion webpages. A significant challenge for developing large scale recommendation systems remains the challenge of obtaining reliable quality and relevance labels for training these systems. While human judges generally provide high quality relevance labels, human labeling is costly, time-consuming and can often be inconsistent. It is difficult to scale to a massive scale, constantly changing recommendation system using solely human labels. This holds true for both training as well as evaluation of the system.

Many products in industry leverage implicit feedback, such as user clicks, as a valuable signal for optimizing recommendation systems. However, excessive focus on click metrics can sometimes result in a preference for clickbait and low-quality recommendations [6]. This issue is particularly critical for webpage recommendation systems due to the significant variability in document quality across the web. Historically, our system has been trained primarily on user interaction and behavior signals. Over time, we had observed a recurring theme in Bing user dissatisfaction feedback submitted for our feature, highlighting concerns about clickbait and low-quality webpage recommendations.

For an index of our size, it is infeasible to curate webpages that enter our index and we thus ran the risk of exposing users to spammy, clickbait or even inappropriate webpages. LLMs have demonstrated an emergent ability to understand user intents and capture semantic relationships between user intents and document content. Recent research has demonstrated that LLMs can accurately comprehend web searcher preferences and generate high-quality relevance labels [11]. Motivated by this, we have developed RecoDCG, an LLM-powered offline recommendation quality metric that facilitates rapid evaluation of new techniques. We further use these quality labels to make our ranking stage quality-aware.

We start with a brief overview of our production webpage-to-webpage recommendation system and illustrate our use of LLMs for developing a reliable quality evaluation metric RecoDCG, providing a playbook of the process we followed. We provide an in-depth discussion of our quality-aware ranking system in Section 4 and elaborate on our application of LLMs for complementary candidate generation in Section 5.

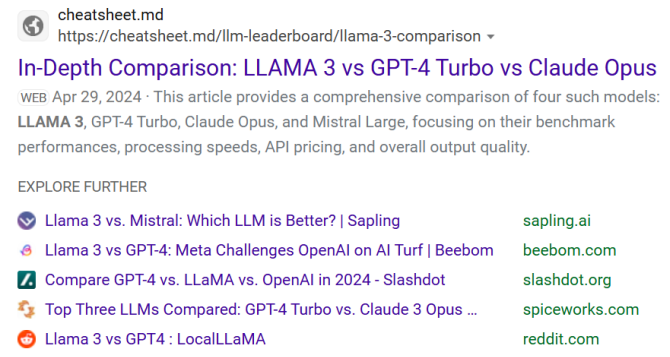


Figure 1: Trigger URL (URL1) with its generated webpage recommendations underneath Explore Further

2 System Overview

The key product objective for our webpage recommendations feature is to provide users with interesting, complementary, and relevant documents that are tied to the URLs appearing in the first 3 positions of search results. The best webpage recommendations either satisfy the user's next query intent, offer an alternative perspective, or introduce an exploratory topic related to URL1. We'll use the term URL1 for the primary (trigger) webpage and URL2 for a recommendation candidate. In Figure 1, the *cheatsheet.md* is the URL1 and the recommended webpages under Explore Further are considered as URL2s.

These recommendations depend solely on the URL1 and are agnostic to the user as well as the user's search query. In the recall stage (candidate selection), we generate relevant candidates tapping into Bing's index of several hundred billion web documents. Our system has multiple recall paths: co-click, collaborative filtering, a two-tower embedding model, as well as our new LLM powered recall pass that we will detail in Section 5. As illustrated in Figure 2, candidate webpages are aggregated and sent to our two layer ranking stage.

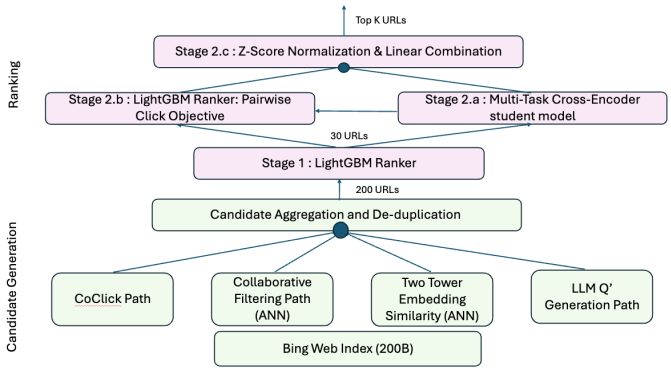


Figure 2: Bing Webpage Recommendations System Diagram

2.1 Candidate Generation Stage Overview

Our production system has multiple candidate generators, referred to as recall paths. Each recall path is tuned to produce a specific number of candidates from Bing's 200 billion webpage index. These candidates are aggregated and de-duplicated before the ranking stage. We define a co-click (co-occurrence click) between two webpages as when multiple Bing user browsing sessions include clicks on both, indicating a degree relatedness. Building a graph from co-click information, we learn webpage embeddings using collaborative filtering (CF) at the scale of billions of webpages. We train a two-tower 6-layer XLM-Roberta [4] based transformer model to learn embeddings for webpages using click logs as labels, where each tower processes the URL and its associated text features. At inference time for a trigger URL1, for both the CF and two-tower model recall paths, we retrieve similar webpages with Approximate Nearest Neighbor (ANN) search based on embedding cosine similarity [5]. In this paper, we introduce a new LLM-powered Q' recall path for generate diverse and complementary candidates for hundreds of millions of URL1s. We discuss Q' recall in detail in Section 5.

2.2 Ranking Stage Overview

Our system's ranking stage consists of, in order:

(1) Stage 1: LightGBM ranker with a smaller set of inexpensive features, mainly URL1-URL2 click count and impression count features over different time windows
(2) Stage 2.a: MT RecoLM: A multitask cross-encoder model with two heads that are trained for pairwise click prediction and recommendation quality tasks respectively
(3) Stage 2.b: LightGBM ranker with a larger set of more expensive features, including cross-encoder model head scores. Optimized with a pairwise click prediction objective (LTR)
(4) Stage 2.c: Linear combination between the Stage 2 LightGBM ranker score and the recommendation quality head score

We discuss quality-aware ranking in detail in Section 4.

3 RecoDCG: An LLM-powered WebPage Recommendation Quality Metric

Motivation Relevance and quality labels from human judges are expensive and slow to obtain. LLMs have been shown to be strong and consistent relevance judges, often outperforming human labelers [11]. For a recommendation system like ours, new items (webpages) are constantly entering the index and we cannot afford to crowd-source human judgements each time we scrape our system. For experiments like new modeling techniques or general system improvements, we needed an inexpensive, fast and robust offline quality metric that we could use to compare a new experiment to the production system. Through several cycles of prompt iteration and metric validation, we developed RecoDCG (Recommendation Discounted Cumulative Gain), an LLM-powered quality metric designed and tuned to be sensitive to the quality of URL2 as a webpage recommendation for URL1. Through our metric development playbook and our subsequent success with productionizing quality improvement, RecoDCG illustrates how LLMs can provide a powerful lever through which we can drive a recommendation system towards a target product direction. In order to develop an LLM-powered metric that accurately measures "the recommendation quality of a URL2 as a recommendation for URL1", we first developed a golden label set of URL1-URL2 pairs that was labelled

by human judges with a comprehensive and clear product guideline. We utilize this golden label set to evaluate different point-wise quality scoring prompts.

3.1 Golden Label Set G

We collected URL1-URL2 pairs from various sources:

(1) Pairs sampled from logs across languages and markets.
(2) Pairs obtained from Bing user dissatisfaction reports.
(3) Reports of search bugs from third-party vendors.
(4) Hard Pairs: Pairs selected to exhibit disagreement between click rate and the quality label from a simple baseline quality prompt: high click rate and low recommendation quality, and vice versa.

The collected set underwent blind human labeling (team members trained with detailed labeling guidelines) to assign a binary ground truth label: Good or Bad. Judging URL1-URL2 pairs for recommendation quality can in practice be a challenging and loosely defined task; whenever two human labels disagreed, we updated our labeling guidelines in a group setting and arrived at the final label through majority consensus. There were several such edge cases for which we arrived at a product consensus through this process. This included segments like subpages as well as establishing clear definitions of what met the bar for clickbait and spam.

3.2 Prompt Development and Evaluation

3.2.1 Pairwise Accuracy Metric: This metric evaluates the consistency of predicted orderings with respect to a golden set of labels. It is defined as the ratio of the number of pairs where the predicted order agrees with the golden set order to the total number of pairs where the label order differs. Since PA has a combinatorial definition (involving all pairs), we developed an optimized implementation leveraging matrix operations to ensure efficient computation on large evaluation sets.

The Pairwise Accuracy (**PA**), reported on the golden label set G, is formally defined as:

$$PA = \frac{\sum_{i<j} I((\hat{y}^i < \hat{y}^j) = (y^i < y^j))}{\sum_{i<j} I(y^i \neq y^j)}$$

where:

- $I(\cdot)$ is the indicator function
- $\hat{y}^i, \hat{y}^j$ are the predicted values for items i and j, respectively.
- y^i, y^j are the true labels for items i and j, respectively.

3.2.2 Prompt Candidates and Results: We develop a pointwise prompt for quality scoring and tune the instructions by evaluating prompt candidates using the golden label set G. Prompt instructions and their phrasing is known to have significant difference in performances [17]. We use GPT-4 and evaluate each prompt candidate on its Pairwise Accuracy (PA) on the golden label set.

Table 1 displays a select few prompt candidates with their Pairwise Accuracy (PA). Prompt Candidate A contains label definitions from 0 to 4 with a set of chain of thought instructions [13] and role-play instructions [9] that ask GPT-4 to serve as 5 independent judges and score the recommendation quality from 0 to 4. We average the 5 scores to obtain a float rating from 0 to 4. Candidate B additionally introduces granularity using the log probabilities returned by GPT-4. We obtain the most likely tokens at each position with their corresponding log probabilities. This enables us to compute a probability-weighted average across the label classes. This change alone results in a 2% absolute improvement in PA. For Candidate C, we additionally break down the scoring task by first reasoning about individual recommendation aspects (topicality, location, authority, and spam awareness) first with chain of thought instructions. This approach led to a significant absolute improvement in PA of 10%. This highlights the importance of breaking down a rating task into sub-tasks and having instructions to reason over these sub-tasks independently.

Version	Pairwise Accuracy
Candidate A	0.77
Candidate B (+ use logprobs for score granularity)	0.79
Candidate C (+ reason over recommendation aspects)	**0.87**

Table 1: Prompt candidates and their pairwise accuracies on the golden label set G

3.2.3 RecoDCG (Recommendation Discounted Cumulative Gain): For techniques and any new experiments, we evaluate the quality improvement through RecoDCG@1 and RecoDCG@5 (since we show a maximum of 5 recommendations). RecoDCG resembles the standard Discounted Cumulative Gain, with the differences being that the relevance scores are LLM generated pointwise scores and we normalize with the sum of the log discounts. The general formula for RecoDCG at a particular rank position is given by:

$$\text{RecoDCG@}k = \frac{\sum_{i=1}^{k} rel_i \cdot \text{discount}(i)}{\sum_{i=1}^{k} \text{discount}(i)}$$

Where:

- $\text{discount}(i) = \frac{1}{\log_2(i+1)}$
- k is the position depth.
- rel_i is the LLM-generated quality label (linearly scaled to a range from 0 to 100 by multiplying by 25 since the pointwise prompt outputs a score from 0 to 4)

3.2.4 RecoDCG offline evaluation set for measuring quality: We build a set of several thousand URL1s by sampling from the top 3 Bing search results for a curated diverse set of frequent historical user queries from Bing logs, collected across markets and languages. To evaluate any new candidate generation improvement or ranking model technique, we scrape our prouction service for the top 5 ranked webpages for each of the URL1s in this evaluation set and compute the RecoDCG@1 and RecoDCG@5. All RecoDCG metrics reported in this paper are computed by scraping control and treatment on this evaluation set of selected URL1s.

3.3 Conditional Click Through Rate (CTR)

We use conditional CTR as our primary online metric for online experiments, as it is conditioned on our feature being triggered and shown in the Bing impression. Simply defined:

$$CTR = \frac{\text{Impressions with at least 1 click on Explore Further}}{\text{Total Explore Further Impressions}}$$

For brevity, we will refer to this metric as CTR in the rest of the paper.

3.4 ClickNDCG: Offline Click Prediction Innerloop

To enhance experimentation agility amidst the finite traffic available for online A/B tests, we evaluated experiments using an offline ranking metric, ClickNDCG. This metric serves as an inner loop for online click-through rate (CTR) optimization. A recommendation system's production logs suffer from both position and selection bias [2]. To collect unbiased logs for an offline evaluation set, we conducted a 2-week random online experiment on a small percentage of traffic. Users were shown a random set of 5 webpages from the 30 candidates that reach the stage 2 LightGBM Ranker. We retain logs with at least one *satisfied* click, defined as a click with a dwell time over a chosen threshold. We call this the ClickNDCG evaluation set and we measure NDCG, assigning a relevance score of 100 to URL2s with a satisfied click and 0 otherwise. The choice of 100 is simply to make the NDCG numbers more interpretable. We validated through several experiments that ClickNDCG@5 aligns with online CTR and is sufficiently sensitive such that improvements or regressions in ClickNDCG@5 translate to movements on online CTR.

3.5 Quality Filtered ClickNDCG: Quality Scores to remove Spurious and Low-Quality Clicks

Not all clicks are equal. We determine a LLM quality score threshold that flags low-quality webpages at high precision. Using a threshold of 1 (on the label scale 0 to 4) to filter out bad recommendation candidates has a precision of 0.87 on the golden label set. Thus, we use a threshold of 1 to exclude clicks on poor quality URL2s from our ClickNDCG evaluation set, removing all impressions where the clicked URL2 had a quality score of 1 or lower. These clicks typically correspond to low-quality, irrelevant, or duplicate webpages. The quality filtered ClickNDCG offline innerloop metric helps provide a clearer picture between the trade-off between quality improvement and clicks worth retaining.

4 LLM-enabled Quality-aware Ranking

4.1 RecoLM: A Multi-Objective Cross-encoder Model

In this section, we describe MT RecoLM, a multitask MiniLM-based cross-encoder model. MT RecoLM is the most powerful model in our ranking stack. As a cross-encoder model, MT RecoLM consumes the text features for a URL1-URL2 pair. In the following sections, we discuss the teacher model pretraining stage, teacher fine-tuning stage and our teacher-student knowledge distillation experiments.

4.2 Teacher Pre-training and Domain Adaptation

We use a 24-layer TuLRv3 [3] model as our base teacher model. For domain adaptation to web search, a large corpus of Bing search logs with billions of impressions is collected. The model is pretrained with the Masked Language Modeling objective as well as ranking objectives: pairwise dwell time prediction and pairwise logged URL position prediction. The input to the model during pretraining is <Query, URL, Title, Snippet> as the goal is to adapt this model to web search as this model is used by several different teams across Bing organization.

4.3 Click Teacher

On top of the pre-trained model, the click teacher has a click head consisting of a few fully connected layers with a single logit output. To adapt the pre-trained model for webpage recommendation click prediction, we fine-tune using click logs specifically from the Explore Further (Web Recommendations) feature. During fine-tuning, the input to the model is <URL1, Title1, Snippet1, URL2, Title2, Snippet2> where Title1 is the webpage title for the trigger URL1, Title2 is the title for URL2 and so on. For generating training data from the logs, we construct triplets consisting of the trigger URL1, a clicked URL2$^+$, and a non-clicked URL2$^-$ from the same impression. We fine-tune on about 1 billion such triplets over a several month window. We fine-tune the entire model and use softmax pairwise loss for this task, formulated below.

$$\text{Pairwise Softmax Loss} = \log\left(\frac{e^{s_{\text{url1\_url2}^+}}}{e^{s_{\text{url1\_url2}^+}} + e^{s_{\text{url1\_url2}^-}}}\right) - \log\left(\frac{e^{s_{\text{url1\_url2}^-}}}{e^{s_{\text{url1\_url2}^+}} + e^{s_{\text{url1\_url2}^-}}}\right) \quad (1)$$

Where:

- $s_{\text{url1\_url2}^+}$ is the model score for the positive (clicked) url2.
- $s_{\text{url1\_url2}^-}$ is the model score for the negative (not-clicked) url2.

During fine-tuning, we apply importance weighting to the loss for each training instance (URL1, URL2_clicked, URL2_not_clicked). The importance weight for each instance is calculated as the logarithm of the frequency of the triplet in the collected training logs. Specifically, if a triplet appears f times in the logs, its importance weight is $\log(f)$. The importance weight increases the contribution of frequent triplets, as triplets with a high f in the logs are naturally more represented in production traffic. Our product has a head-heavy distribution of URL1s, with a small percentage of popular URL1s garnering the large portion of impressions. The logarithm operation helps address the imbalance and tapers the influence of highly frequent instances, allowing less frequent triplets to have a more significant impact on the loss function.

4.4 Recommendation Quality Teacher

On top of the pretrained model, the quality teacher has a classification head with 5 logits corresponding to the quality classes 0 to 4. We develop a robust labeling pipeline capable of handling large volume and use it for obtaining quality scores (range from 0 to 4) for 80 million URL1-URL2 pairs with GPT-4. For fine-tuning the relevance teacher, we experiment with both cross entropy and soft cross entropy loss. For cross entropy (CE) loss, we round the quality scores to the nearest integer, while to obtain soft target probabilities for soft cross entropy (SCE) Loss, we assign probability mass to the two closest integer classes.

The soft cross entropy (SCE) loss is defined as:

$$L_{\text{SCE}} = -\sum_{i=1}^{C} p_i \log(q_i)$$

where:

- C is the number of classes, 5.
- p_i is the soft target probability for class i 1
- q_i is the model's predicted probability for class i

As Table 2 illustrates, fine-tuning the teacher model with CE loss shows marginally better RecoDCG@5 gain compared to SCE loss. The numbers in the table are both reported as deltas compared to using the baseline model (Click Teacher) for ranking.

Model	RecoDCG@5
CE Loss	+8.48 points
SCE Loss	+8.29 points

Table 2: Relevance teacher fine-tuning loss: SCE vs CE

4.5 MT RecoLM: Knowledge Distillation

As our product appears on the Bing search results page, our ranking stage has hard latency requirements. In order to serve online, we need to distill to a much smaller cross-encoder model. For knowledge distillation, we collect about 1 billion (URL1,URL2) pairs from a 3 month window and stamp this set with both the click and quality teachers. We then distill to a 6-layer student model based on MiniLM [12], with two heads: one for pairwise click prediction and the other for recommendation quality classification. We coined this model the MT (Multi-task) RecoLM student model. Both heads are distilled using MSE (Mean Squared Error) loss. Due to the quality head having 5 logits compared to the click head's 1 logit, we observed a performance imbalance. To address this, we adjusted the loss weights of the distillation losses, assigning a larger weight to the click head distillation loss to maintain comparable ClickNDCG@5 performance. We do continual training: we initialize the model weights with a checkpoint from the previous productionized click-prediction only model.

In Table 3, we show how our teacher-student knowledge distillation with the 1 billion (URL1,URL2) distillation data fares against directly fine-tuning the student model with SCE loss for the quality classification task on the 80 million GPT-4 labelled data. We show that distillation demonstrates a significant gain of over 8 points of RecoDCG@5 compared to direct fine-tuning. This highlights the important role of distillation for this task when training multi-objective cross-encoder models for recommendations, even for web-scale data.

In Table 3, RecoDCG@5 is reported by ranking webpages by the quality head scores of the MT RecoLM student model. Both approaches are compared against the baseline single objective student model (optimized solely for clicks) performance.

4.6 Final Ranking Score

To utilize the $0-4$ recommendation quality score from our multi-task transformer model online, we sought an adjustable method to influence the final ranking scores of candidate webpages. Conducting online A/B flights for various linear and non-linear combination strategies is impractical. Hence, we leverage the aforementioned offline evaluation metrics to identify promising settings for online tests. In the next section, we discuss how the final ranking score is a combination of the stage 2 LightGBM click ranker score and the MT RecoLM quality head score and how the combination weight is determined.

Model	RecoDCG@5
Teacher Distillation (MSE Loss)	+9.76
Direct Fine-tuning (SCE Loss)	+1.44

Table 3: MT RecoLM student model: Teacher Distillation vs Fine-tuning

4.7 Trade-off between clicks and recommendation quality

From several online experiments, we observe clicks and quality are often misaligned. Frequent cases include URL2s with duplicate content, clickbait titles, location specific URL2s that redirect to the same URL1 that get clicks due to click shift, spam URL2s, and irrelevant yet popular webpages.

In order to inform a deliberate trade-off between CTR and RecoDCG as well to decide on a score combination strategy, we evaluate by plotting the trade-off curve of quality filtered ClickNDCG@5 (discussed in section 3.5) and RecoDCG@5 on our offline evaluation sets. We experiment with several combination strategies including dynamic weighting based on the magnitude of the quality score as well as hand-crafted piecewise functions. However, a linear combination of z-score normalized scores yielded the optimal trade-off curve.

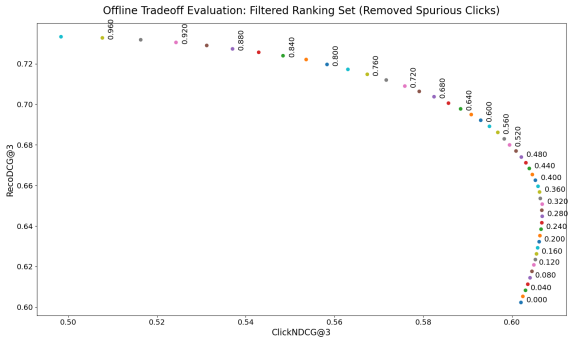


Figure 3: Trade-off curve: RecoDCG@5 vs quality filtered ClickNDCG@5

4.8 Linear Combination

After deploying the MT RecoLM student model discussed in the earlier section, we also re-trained our Stage 2 LightGBM Ranker after feeding the MT RecoLM student model's click head and quality head scores as features. The LightGBM ranker score and the multi-task student model's quality head scores are first Z-score normalized

using the mean and standard deviation computed over a sampled set of logs that represent production traffic. We deploy a linear combination between the stage 2 LightGBM ranker score and the quality head score as:

$$(1-\lambda)\cdot\frac{\text{lgbm\_score}-\mu_{\text{lgbm\_score}}}{\sigma_{\text{lgbm\_score}}}+\lambda\cdot\frac{\text{cls\_score}-\mu_{\text{cls\_score}}}{\sigma_{\text{cls\_score}}}$$

- λ: quality weight
- cls_score: recommendation quality head score
- lgbm_score: stage 2 LightGBM ranker score

From the offline evaluation curve shown in Figure 3, a quality weight (λ) of 0.32 was chosen as the right-most point on the tradeoff curve. It both maximizes the quality filtered ClickNDCG@5 and provides a large quality improvement as measured through a +5.10 RecoDCG@5 improvement over the production baseline.

5 LLMs for Candidate Generation: Q' Recall

In this section, we detail our LLM-powered Q' recall path. We harnessed LLMs to create a new recall path for generating relevant and interesting webpage recommendation candidates that are complementary to those returned by other recall paths.

The overall architecture of this recall path's offline computation and online flow is shown in Figure 4.

Offline Pipeline: For each request for a URL1, we employ an LLM to generate queries Q' related to the content of the webpage, that are complementary and relevant enough to yield high-quality webpage recommendations. Subsequently, we utilize Bing search to retrieve the top 10 URLs for each generated query. We then de-duplicate the combined retrieved URL set and filter out low relevance webpages. For each URL1, we store an array of upto 30 URL2 candidates in a highly performant key-value store.

Online Serving: During online serving, for each user impression on a trigger URL1, we query the the key-value store to fetch the URL2 candidate array. These candidates are aggregated with those from the other recall paths and sent to the ranking stage.

5.1 Q' Generation Prompt

For generating queries Q' from a given URL, we leverage the content of the webpage, including as input the URL, title, and the webpage body (obtained by processing and cleaning the webpage HTML). We extensively tuned our prompt to arrive at a version that can consistently generate a relevant yet complementary set of queries given a webpage. In developing the query generation prompt, we employed prompting methods such as few-shot prompting [15, 10] and chain-of-thought (CoT) [1]. Aside from instructions tailored to our specific problem, below are some key learnings:

- We found it critical to include non-English few-shot examples in the prompt and to explicitly instruct the model to generate queries in the document's language in order to avoid language mismatch.
- Instead of instructing the prompt to generate for a specific number of queries apriori, prompting the model to stop when query diversity decreases. The average number of queries generated per URL is roughly 6.
- Instructing the model to score each generated query based on relevance to the webpage, allowing us to discard queries with limited relevance in post-processing.

In Table 4, we present a collection of queries generated by GPT-4 related to Adele's widely popular song "Hello". These queries exhibit diversity and cover various aspects of the song as well as the artist that a web user may be interested in.

5.2 From Q' generated queries to Candidate URLs

Using the generated queries Q' for a given URL1, we leverage Bing web search as the retrieval engine to obtain the top 10 URLs for each Q' query. We invested heavily in developing and optimizing a extremely high throughput service to scrape Bing production at scale for our use case to build a several hundred million Q' recall index. Subsequently, post-processing filters for spam and de-duplication are applied to the scraped URLs.

GPT-4 Generated Queries
1- Adele's new album 30 and its lead single Easy On Me
2- The meaning and aspiration behind Adele's Hello song
3- The collaboration and production of Adele and Xavier Dolan for the Hello music video
4- Adele's musical style and influences over the years
5- Adele's fan reactions and comments on Hello music video
Mistral-7B Generated Queries
1- Adele's comeback with 30 album and Easy On Me single
2- The meaning and inspiration behind Hello by Adele
3- The making of Hello music video by Xavier Dolan
4- Adele's best songs and performances

Table 4: GPT-4 and Mistral-7B generated queries for Adele's Hello song page on YouTube.

5.3 GPT-4 powered Q' Recall Path

Whilst designing our pipeline, to validate the potential of our approach we used GPT-4 for query generation. We scaled the recall path's index size to approximately 100 million documents. This method demonstrated end-to-end improvements on both RecoDCG@5 as well as online CTR as shown in Table 5. The production outcomes in Table 5 are all computed against the baseline of the previous production system prior to introducing these LLM techniques for our system.

5.4 Scaling by Supervised Fine-tuning a Small Language Model (SLM)

A major challenge inherent to having an large index is that inference of GPT-4 sized LLMs is prohibitively expensive and slow. For iterating on the query generation prompt and then generating training data we use GPT-4. However, scaling our Q' recall path to billions of webpages is crucial for us to enhance candidate selection meaningfully across the index. Therefore, to achieve scalability,

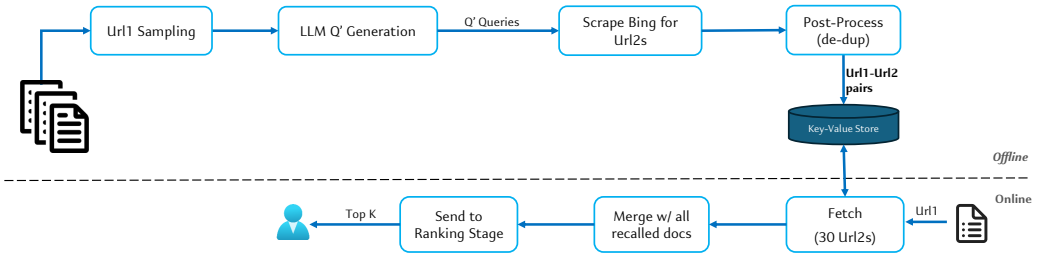


Figure 4: Architecture of LLM Q' generation recall path. By *online* we refer to situations when our recommendation systems is actively engaged with real-time data and user interactions. By *offline* we refer to scenarios where our system is not directly interacting with real-time data or users.

we turned to Small Language Models (SLMs) such as Mistral-7B. Although SLMs may offer lower performance compared to GPT-4 sized models in a few-shot setting, they provide significantly greater throughput and can be fine-tuned and tailored to a specific task with better performance [16, 14].

Q' Training Data Generation: Using GPT-4 and our tuned prompt, we generate a set of Q' queries for all 2 million URL1s. We implement post-processing logic to ensure consistent quality of the list of generated queries such as filters on length and language.

Supervised Fine-tuning (SFT): We fine-tune Mistral-7b with SFT using GPT-4 generated Q' queries as targets and the query generation prompt with the webpage features as input. For the finetuning experiments, we use a cluster with 48 Tesla V100-32GB GPUs. The fine-tuned Mistral-7B maintains comparable generation quality whilst increasing our throughput by over 60x on the same hardware compared to GPT-4, enabling us to scale to a 600 million Q' recall index at a fraction of the compute cost. Table 5 demonstrates the parity performance of using a fine-tuned Mistral-7B versus using GPT-4 as the Q' generator, evaluating the respective recall paths on RecoDCG@5. For fair comparison, for RecoDCG@5 we report all numbers in 5 on the same triggered subset of URL1s from the RecoDCG offline evaluation set defined in Section 3.2.4.

The SFT objective is a classification task over the vocabulary V, using cross-entropy loss:

$$L = -\sum_{i=1}^{V} y_i \log(\hat{p}_i)$$

- V is the size of the vocabulary.
- y_i is the i-th element of the true token vector (one-hot encoded).
- $\hat{p}_i$ is the predicted probability of the i-th token in vocabulary.

We use the Adam optimizer [8] for weight updates and set the learning rate to 1e-5.

Model Overview: Across several SLMs we fine-tuned as Q' generators, we tested the performance both online (CTR) and offline. Mistral-7B, released in Sep 2023, proved to be the most performant model for the task. Mistral-7B is a decoder only model with 7.3B parameters. It uses Sliding Window Attention (SWA), is trained with 8k context length and uses Grouped Query Attention for faster inference- making it a good fit for fast, web-scale throughput [7].

Table 4 shows examples of the queries generated by fine-tuned Mistral-7B model compared to those generated by GPT-4 for the YouTube webpage for Adele's song *Hello*.

Evaluation: We present the evaluation results in Table 5, demonstrating that a recall path constructed by fine-tuning Mistral-7B for the Q' task achieves comparable performance to a recall path with GPT-4, as measured by RecoDCG@5. We obtain a inference throughput of 60x compared to GPT-4 on the same hardware, enabling us to scale to a significantly larger index for the Q' recall path. This drives the online CTR gain of Mistral-7B powered recall path compared to the baseline version powered by GPT-4 in Table 5. For thoroughness, we conduct an online A/B experiment with the index size of the Mistral-7B powered recall path matched to that of the GPT-4 powered recall path. The results of this experiment demonstrate parity in both online click-through rate (CTR) and RecoDCG@5 metrics.

5.5 Mistral-7B Enhanced: Refined Training Data Through Implicit User Feedback (Clicks)

Our experimentation has consistently underscored the critical role of high-quality data in fine-tuning models. To improve the quality of our training data, we decided to integrate implicit user feedback i.e. clicks. We conduct several online experiments wherein we surface fixed URL1-URL2 recommendations generated from the full GPT-4 generated Q' training data and log user clicks and interactions. We then refine our training data by retaining only those queries where the corresponding URLs received a satisfied click (having a dwell time over a chosen threshold). As a result, we filtered out each query associated with the candidate URLs that were not sufficiently interesting or relevant enough to receive any clicks, thereby adapting the training data for our task leveraging implicit user feedback.

We fine-tune Mistral-7B again on this refined dataset and call it Mistral-7B Enhanced. We evaluate this approach with an online A/B test and find that it demonstrates promising improvements in both Online CTR and RecoDCG@5, as detailed in Table 5. Due to current resource constraints, we have not yet deployed a massive-scale Q' recall index powered by Mistral-7B Enhanced but the demonstrated gains through this approach may warrant the necessary investment.

	A/B Test	Online CTR	RecoDCG@5
1	Quality-Aware Ranking	-1.20%	+5.100
2	Q' Recall (GPT-4)	+0.89%	+0.760
3	Q' Recall (Mistral-7B)	+1.71%	+0.762
4	Q' Recall (Mistral-7B Enhanced)	+1.99%	+0.832
5	**Deployed Treatment (E2E)**	**+0.52%**	**+5.862**

Table 5: Online A/B results and RecoDCG@5 reported with the baseline system as control for all rows. The deployed treatment E2E (End-to-End) refers to the combined deployment of both quality-aware ranking (row 1) and Q' recall Mistral-7B (row 3) in the candidate generation stage. The online CTR improvement of Mistral-7B compared to GPT-4 is coming from the 6x larger index size.

6 Results and Metrics

For the techniques discussed in both the ranking and candidate generation sections, we provide a set of ablations: online CTR numbers (from A/B tests with the baseline system as control) and offline RecoDCG@5 numbers in Table 5. All CTR and RecoDCG numbers are compared with respect to the baseline system. The deployed quality-aware ranking uses a linear combination weight of (λ) = 0.32, as discussed in Section 4.8. This demonstrates a 1.2% CTR regression with a 5.1 point increase in RecoDCG@5. We validate the CTR regression and explain why we are comfortable making this tradeoff in the following section. The deployed Q' recall path with Mistral-7b gives a 1.71% CTR gain with a 0.762 point improvement in RecoDCG@5. For the combined treatment (Row 1 and Row 3), we report a net 0.52% improvement in CTR and a 5.862 point improvement in RecoDCG@5 over the baseline system. This is a significant quality improvement for our recommendation system that we deployed to production with no latency increase. Our Q' recall path generates candidates through an offline pipeline and the fetch request at inference time runs in parallel with the other recall paths. The multi-task student model deployed in quality-aware ranking maintains the same size as the student model of our baseline system and the additional classification head computes the quality score in parallel with the click head. Thereby, our end-to-end latency remains unaffected.

6.1 Validating the CTR Regression

To analyze the the 1.2% decrease in CTR for Quality-Aware Ranking, we conducted the following analysis using our online experiment logs. We collected data by running multiple online experiments with varying the linear combination weight (λ) from 0.1 to 0.5. We then gathered (URL1, URL2_a, URL2_b) triplets where the LLM-generated quality label and online click counts disagreed the most. Subsequently, we conducted a blind review with human judges (team members calibrated with the help of labeling guidelines), presenting the webpage contents and asking them to indicate their preferred candidate between URL2_a and URL2_b for each URL1. The order of presentation was randomized to ensure that judges were blind to which URL2 was associated with the treatment. Each triplet was labeled independently by two judges and if these two judges disagreed, a third judge cast the tie-breaking vote yielding the final human judge label.

Our findings show, with statistical significance, that human judges' preferences align more closely with the quality labels generated by LLMs compared to online click counts. Specifically, human judges' final assessments agreed with the quality label 54% of the time, whereas alignment with click counts occurred only 6% of the time. The remaining 40% of cases were ties, cases where the human judges exhibited indifference between URL2_a and URL2_b. This corroborates that RecoDCG serves as a superior indicator of product quality compared to click counts. Nonetheless, the substantial proportion of ties suggests that there is potential for further refinement of the RecoDCG metric to enhance its precision. Upon examining the triplets, we observe that a large chunk of the lost clicks primarily on URL2s that were duplicate and redirect webpages, subpages, clickbait and webpages with limited relevance. This analysis justified our decision to deploy to the production.

6.2 Improved Product Experience

In Figure 5, we present an example in which the deployed quality improvement (Treatment T1) leads to a considerably improved product experience over the baseline system (Control C). The URL1 is a link to the RFM station on radio.net. For this URL1, there is a significant difference in RecoDCG@5 between control and treatment, and thus a large quality difference in the top 5 recommendations shown. Several of the URL2s in the control slate redirect to URL1 and offer limited value. In our deployed treatment shown on the right under T1, these are replaced with improved recommendations, offering a more diverse range of alternative online radio services. Over the course of the past few months since we have deployed the treatment to production, we have seen a marked drop in user dissatisfaction feedback reports from Bing users especially around duplicate, clickbait/low-quality and low relevance content.

7 Conclusion

We demonstrate how a recommendations system can use LLMs to develop and validate a reliable offline quality metric that can serve as an powerful lever to measure improvement and drive product direction. We present our approach to leveraging LLM-generated quality labels to do quality-aware ranking and discuss the evaluation and trade-off between quality and clicks. For candidate generation, we highlight our effective use of LLMs to generate complementary and diverse recommendation candidates offline. We show that fine-tuning a SLM enables us to scale to significantly larger index sizes without compromising Q' query generation quality. Our overall deployed technique achieves a significant recommendation quality gain as well as CTR increase in production, without any meaningful increase in online costs or latency.

Given the inherent noise in implicit user feedback, particularly clicks, for future work we plan to leverage high throughput fine-tuned SLMs to remove spurious, low-quality clicks from the training data for our ranking models. This helps prevent our ranking models from generalizing unwanted behavior learned from user clicks such as an inclination for clickbait titles. We plan to apply the same idea to our recall paths: cleaning up and augmenting the training data for our large-scale collaborative filtering training and for the two-tower dense retrieval model.

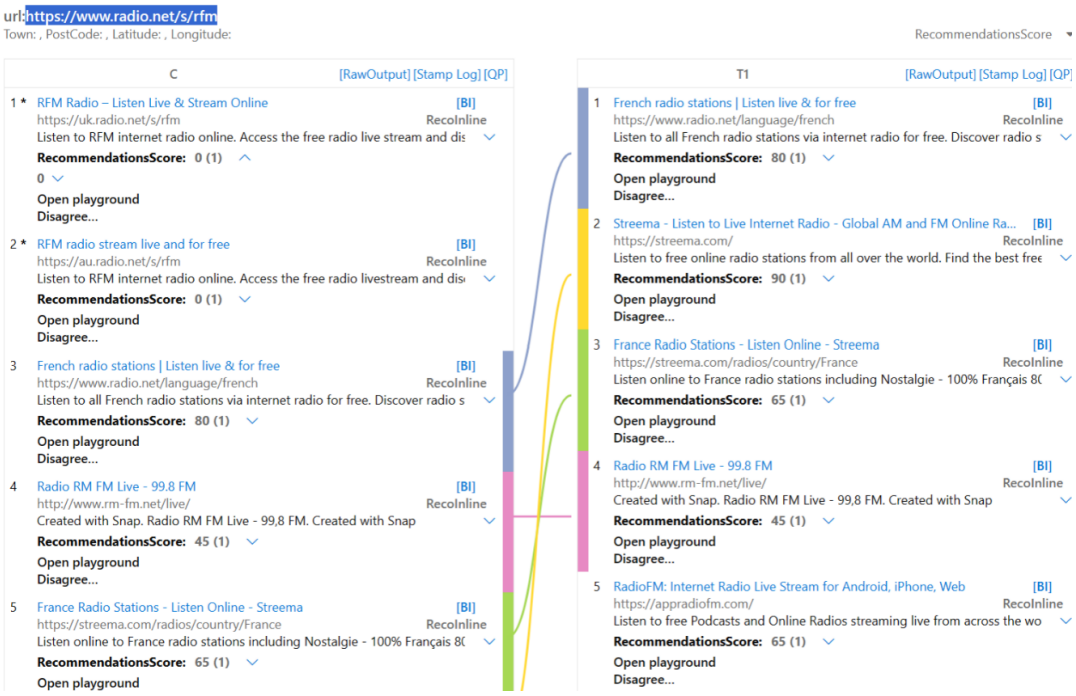


Figure 5: Example highlighting the improved product experience, URL1: *https://www.radio.net/s/rfm*

References

[1] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[2] Jiawei Chen et al. *Bias and Debias in Recommender System: A Survey and Future Directions*. 2021. arXiv: 2010.03240 [cs.IR]. URL: https://arxiv.org/abs/2010.03240.

[3] Zewen Chi et al. *InfoXLM: An Information-Theoretic Framework for Cross-Lingual Language Model Pre-Training*. 2021. arXiv: 2007.07834 [cs.CL].

[4] Alexis Conneau et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL].

[5] Matthijs Douze et al. *The Faiss library*. 2024. arXiv: 2401.08281 [cs.LG]. URL: https://arxiv.org/abs/2401.08281.

[6] Nicole Immorlica, Meena Jagadeesan, and Brendan Lucier. *Clickbait vs. Quality: How Engagement-Based Optimization Shapes the Content Landscape in Online Platforms*. 2024. arXiv: 2401.09804 [cs.GT].

[7] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

[8] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[9] Aobo Kong et al. *Better Zero-Shot Reasoning with Role-Play Prompting*. 2024. arXiv: 2308.07702 [cs.CL]. URL: https://arxiv.org/abs/2308.07702.

[10] Fina Polat, Ilaria Tiddi, and Paul Groth. "Testing Prompt Engineering Methods for Knowledge Extraction from Text". In: *Semantic Web. Under Review* (2024).

[11] Paul Thomas et al. *Large language models can accurately predict searcher preferences*. 2024. arXiv: 2309.10621 [cs.IR]. URL: https://arxiv.org/abs/2309.10621.

[12] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: 2002.10957 [cs.CL].

[13] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.

[14] Xiaohan Xu et al. "A survey on knowledge distillation of large language models". In: *arXiv preprint arXiv:2402.13116* (2024).

[15] Shengyu Zhang et al. "Instruction tuning for large language models: A survey". In: *arXiv preprint arXiv:2308.10792* (2023).

[16] Justin Zhao et al. *LoRA Land: 310 Fine-tuned LLMs that Rival GPT-4, A Technical Report*. 2024. arXiv: 2405.00732 [cs.CL]. URL: https://arxiv.org/abs/2405.00732.

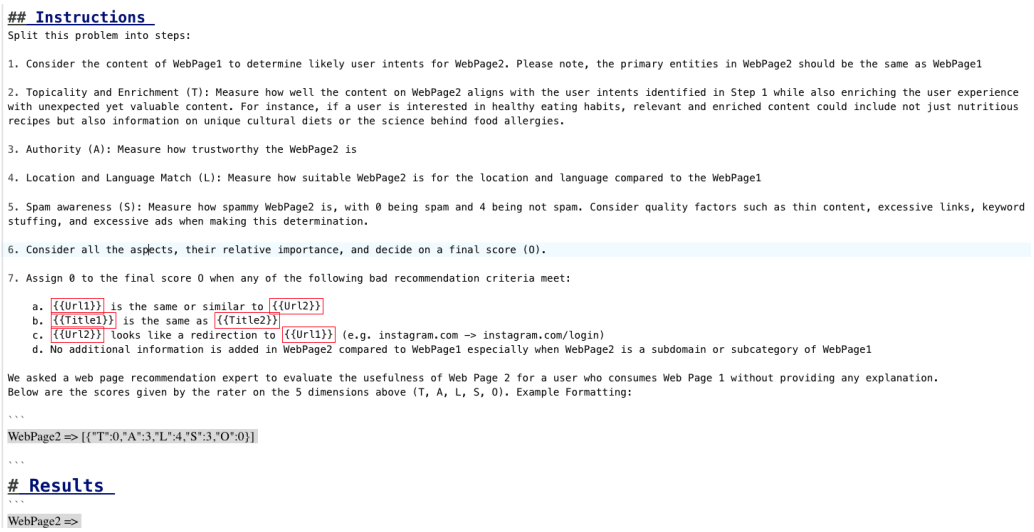
[17] Yongchao Zhou et al. *Large Language Models Are Human-Level Prompt Engineers*. 2023. arXiv: 2211.01910 [cs.LG]. URL: https://arxiv.org/abs/2211.01910.

8 Appendix

Algorithm 1 Convert a float Quality Score to a soft label for Soft Cross Entropy Loss

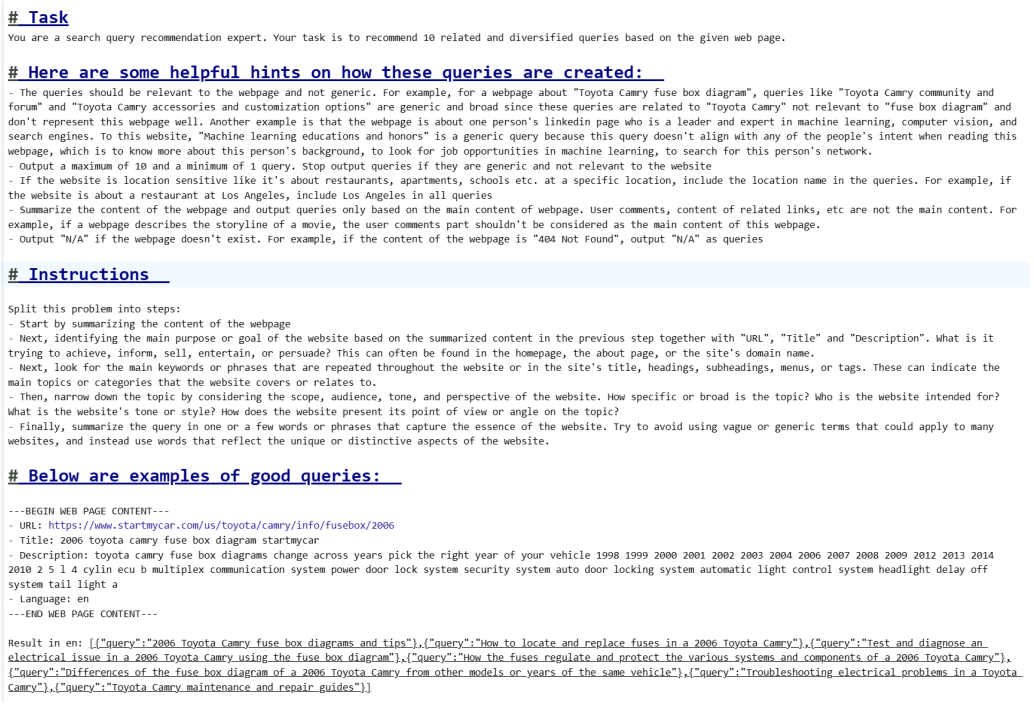
Input: $label$
$sce\_labels \leftarrow [0, 0, 0, 0, 0]$
$lowindex \leftarrow label\%1$
$highindex \leftarrow lowindex + 1$
$diff \leftarrow label - (label\%1)$
$sce\_labels[lowindex] \leftarrow (1 - diff)$
$sce\_labels[highindex] \leftarrow diff$
Output: sce_labels

Received 2024; revised 2024; accepted 2024



```
## Instructions
Split this problem into steps:

1. Consider the content of WebPage1 to determine likely user intents for WebPage2. Please note, the primary entities in WebPage2 should be the same as WebPage1

2. Topicality and Enrichment (T): Measure how well the content on WebPage2 aligns with the user intents identified in Step 1 while also enriching the user experience
with unexpected yet valuable content. For instance, if a user is interested in healthy eating habits, relevant and enriched content could include not just nutritious
recipes but also information on unique cultural diets or the science behind food allergies.

3. Authority (A): Measure how trustworthy the WebPage2 is

4. Location and Language Match (L): Measure how suitable WebPage2 is for the location and language compared to the WebPage1

5. Spam awareness (S): Measure how spammy WebPage2 is, with 0 being spam and 4 being not spam. Consider quality factors such as thin content, excessive links, keyword
stuffing, and excessive ads when making this determination.

6. Consider all the aspects, their relative importance, and decide on a final score (O).

7. Assign 0 to the final score O when any of the following bad recommendation criteria meet:

    a. {{Url1}} is the same or similar to {{Url2}}
    b. {{Title1}} is the same as {{Title2}}
    c. {{Url2}} looks like a redirection to {{Url1}} (e.g. instagram.com -> instagram.com/login)
    d. No additional information is added in WebPage2 compared to WebPage1 especially when WebPage2 is a subdomain or subcategory of WebPage1

We asked a web page recommendation expert to evaluate the usefulness of Web Page 2 for a user who consumes Web Page 1 without providing any explanation.
Below are the scores given by the rater on the 5 dimensions above (T, A, L, S, O). Example Formatting:

```
WebPage2 => [{"T":0,"A":3,"L":4,"S":3,"O":0}]
```

Results
```
WebPage2 =>
```
```

Figure 6: Final RecoDCG Quality Scoring Prompt Excerpt (Candidate C) with chain of thought instructions for recommendation dimensions, given <URL1,Title,Snippet1,URL2,Title2,Snippet2>



```
# Task
You are a search query recommendation expert. Your task is to recommend 10 related and diversified queries based on the given web page.

# Here are some helpful hints on how these queries are created:
- The queries should be relevant to the webpage and not generic. For example, for a webpage about "Toyota Camry fuse box diagram", queries like "Toyota Camry community and
forum" and "Toyota Camry accessories and customization options" are generic and broad since these queries are related to "Toyota Camry" not relevant to "fuse box diagram" and
don't represent this webpage well. Another example is that the webpage is about one person's linkedin page who is a leader and expert in machine learning, computer vision, and
search engines. To this website, "Machine learning educations and honors" is a generic query because this query doesn't align with any of the people's intent when reading this
webpage, which is to know more about this person's background, to look for job opportunities in machine learning, to search for this person's network.
- Output a maximum of 10 and a minimum of 1 query. Stop output queries if they are generic and not relevant to the website
- If the website is location sensitive like it's about restaurants, apartments, schools etc. at a specific location, include the location name in the queries. For example, if
the website is about a restaurant at Los Angeles, include Los Angeles in all queries
- Summarize the content of the webpage and output queries only based on the main content of webpage. User comments, content of related links, etc are not the main content. For
example, if a webpage describes the storyline of a movie, the user comments part shouldn't be considered as the main content of this webpage.
- Output "N/A" if the webpage doesn't exist. For example, if the content of the webpage is "404 Not Found", output "N/A" as queries

# Instructions

Split this problem into steps:
- Start by summarizing the content of the webpage
- Next, identifying the main purpose or goal of the website based on the summarized content in the previous step together with "URL", "Title" and "Description". What is it
trying to achieve, inform, sell, entertain, or persuade? This can often be found in the homepage, the about page, or the site's domain name.
- Next, look for the main keywords or phrases that are repeated throughout the website or in the site's title, headings, subheadings, menus, or tags. These can indicate the
main topics or categories that the website covers or relates to.
- Then, narrow down the topic by considering the scope, audience, tone, and perspective of the website. How specific or broad is the topic? Who is the website intended for?
What is the website's tone or style? How does the website present its point of view or angle on the topic?
- Finally, summarize the query in one or a few words or phrases that capture the essence of the website. Try to avoid using vague or generic terms that could apply to many
websites, and instead use words that reflect the unique or distinctive aspects of the website.

# Below are examples of good queries:

---BEGIN WEB PAGE CONTENT---
- URL: https://www.startmycar.com/us/toyota/camry/info/fusebox/2006
- Title: 2006 toyota camry fuse box diagram startmycar
- Description: toyota camry fuse box diagrams change across years pick the right year of your vehicle 1998 1999 2000 2001 2002 2003 2004 2006 2007 2008 2009 2012 2013 2014
2010 2 5 l 4 cylin ecu b multiplex communication system power door lock system security system auto door locking system automatic light control system headlight delay off
system tail light a
- Language: en
---END WEB PAGE CONTENT---

Result in en: [{"query":"2006 Toyota Camry fuse box diagrams and tips"},{"query":"How to locate and replace fuses in a 2006 Toyota Camry"},{"query":"Test and diagnose an
electrical issue in a 2006 Toyota Camry using the fuse box diagram"},{"query":"How the fuses regulate and protect the various systems and components of a 2006 Toyota Camry"},
{"query":"Differences of the fuse box diagram of a 2006 Toyota Camry from other models or years of the same vehicle"},{"query":"Troubleshooting electrical problems in a Toyota
Camry"},{"query":"Toyota Camry maintenance and repair guides"}]
```

Figure 7: Q' Queries Generation Prompt Excerpt given <URL1,Title1,Snippet1>