# Query Variant Detection Using Retriever as Environment

**Minji Seo**[1]* **Youngwon Lee**[1]* **Seung-won Hwang**[1]†
**Seoho Song**[2] **Hee-Cheol Seo**[2] **Young-In Song**[2]
[1]Seoul National University [2]NAVER Corp.

## Abstract

This paper addresses the challenge of detecting query variants—pairs of queries with identical intents. One application in commercial search engines is reformulating user queries with its variant online. While measuring pairwise query similarity has been an established standard, it often falls short of capturing semantic equivalence when word forms or order differ. We propose leveraging the retrieval as an environment feedback (EF), based on the premise that desirable retrieval outcomes from equivalent queries should be interchangeable. Experimental results on both proprietary and public datasets demonstrate the efficacy of the proposed method, both with and without LLM calls.

## 1 Introduction

Identifying query variants—semantically equivalent queries—is critical for ensuring search engines consistently return identical results for queries that reflect the same intent. One application of this detection is query reformulation, where a user query $q$ is augmented or replaced with its variant $q'$ to improve quality and consistency in retrieval results.

However, identifying query variants is non-trivial as a highly similar query pair, often relying heavily on lexical similarity between $q$ and $q'$, may fail to differ in word form, order, or phrasing despite sharing the same intent (Iida and Okazaki, 2021).

When latency requirements are relaxed, Large Language Models (LLMs) may offer an improved semantic understanding (Chen et al., 2023), and have been used for query understanding related tasks such as classifying search intent (Srinivasan et al., 2022). LLMs have the advantage of observing query variants in diverse surrounding contexts during pretraining, which allows them to more reliably identify query variants. However, their computational cost makes them impractical for latency-sensitive, real-time applications involving commercial search engines.

Our work demonstrates how leveraging the retriever as an Environment Feedback (EF) enhances query variant detection across diverse scenarios. EF utilizes retrieval results as additional features—by quantifying query-document or document-document similarity—beyond traditional pairwise query similarity. For instance, retrieval results for query variants exhibit high similarity (Ni et al., 2021). Specifically, we show these additional EF features improve performance in both latency-sensitive cases (by training an efficient classifier) and latency-relaxed cases (by integrating with LLMs). Our generalized approach naturally supports public data with limited training annotations, or weaker EF as well.

Our contributions are as follows:

- We designed and trained an efficient classifier that effectively utilizes EF without LLM calls.
- We show that our method, outperforms LLM-only approach, by combining with both stronger and weaker types of EF.
- We release expert annotations to foster future efforts.[1]

## 2 Related Work

This section overviews the task of query variant identification (Section 2.1), and relevant literature on utilizing the environment feedback from retriever (Section 2.2).

### 2.1 Query Similarity and Query Variants

Query variant task is an instance of a broader class of query understanding, used for query clustering

*Equal contribution.
†Correspondence to: seungwonh@snu.ac.kr.

[1]https://github.com/Minji-Seo/NAACL-25-Industry-ManualDataset.git



*Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Industry Track)*, pages 662–671

and query rewriting (Chien et al., 2018; Azhir et al., 2021; Li et al., 2022; Farzana et al., 2023). By organizing related queries or reformulating them, the retrieval quality of search engines can be enhanced. While expert annotation is required, the following pseudo signals have been used as proxy for scaling.

**Lexical Matching** Word overlaps or edit distance quantify lexical similarity (Zhang and Dong, 2002; Li et al., 2006; Gao et al., 2010) as a proxy of pairwise similarity.

**Clicks** Co-clicks, a representative example of a post-search behavior feature, provide a useful signal that hints query similarity and helps distinguishing false positives in lexical matching (e.g., 'SVN' and 'SVM'), often derived from co-clicked URLs or session data (Beeferman and Berger, 2000; Wen et al., 2001; Paredes and Chávez, 2005; Cao et al., 2008). As clicks are collected only from high-ranked results, they are rank-biased.

**Taxonomy** Hierarchical taxonomies (Zhang and Dong, 2002; Farzana et al., 2023) of co-clicked documents provides deeper semantic signals.

### 2.2 Our Distinction

Our distinction is leveraging retriever and LLM as verification signals, and extend to consider query-document (QD) and document-document (DD) relations for verification.

The most well-known form of EF from a retriever is pseudo-relevance feedback (PRF) methods such as Rocchio's or Relevance Model (Lavrenko and Croft, 2003). Top-$k$ results from the retriever are used as a proxy of gold relevance annotations for true query-document relevance, $\mathcal{R}^*(q, d)$. Unlike existing work using the rank as an entangled feedback for a single query, we disentangle the QQ, QD and DD similarities, as described in Section 4.

While incurring additional computational cost, verifiers as proxies or supplements to LLMs have been actively adopted to balance accuracy and efficiency (Chen et al., 2023; Wang et al., 2024). We show this information can enhance verification.

## 3 Preliminaries

We first provide the task formulation and basic notation to be used for the rest of the paper.

### 3.1 Retrieving Top-$k$ Documents

Given a search query $q$ and the corpus of documents $\mathcal{D}$, the goal of the retriever is to surface the set of relevant documents

$$R_q^* = \{d \,|\, d \in \mathcal{D},\, \mathcal{R}^*(q, d) = 1\}, \tag{1}$$

where $\mathcal{R}^*(q, d)$ denotes the underlying true binary relevance label, in its top-$k$ retrieval result $R_q^{(k)}$,

$$R_q^{(k)} = \text{topk}(\mathcal{R}(q, d)), \tag{2}$$

where $\mathcal{R}(q, d)$ is the relevance score the retriever assigned to $d$ with respect to $q$. For the sake of simplicity of notation, we will be referring to $R_q^{(k)}$ as simply $R_q$, as we will consider a fixed $k$ for top-$k$ retrieval throughout the paper's context.

### 3.2 Problem Statement

In this paper, we consider the task of query variants identification, or semantic equivalence classification of deciding whether two given queries $q$ and $q'$ are equivalent. Two queries are considered equivalent, if and only if their relevant document sets are the same, that is,

$$q \sim q' \quad \text{iff.} \quad R_q^* = R_{q'}^*. \tag{3}$$

We consider a basic form of lightweight classifier $f$, or, verifier, that only considers the pairwise query similarity between the two, which can be denoted as

$$\hat{y} = f(q, q'), \tag{4}$$

where $\hat{y}$ is the binary prediction on query equivalence. An LLM verifier $\theta$ can be used in-place as a stronger classifier, with their access to vast parametric knowledge obtained during their pretraining

$$\hat{y} = \texttt{LLM}(q, q'; \theta), \tag{5}$$

at increased inference costs.

## 4 Method

We first discuss the baseline of training the verifier $f$ in a supervised fashion according to Eq. 4, utilizing the queries $q$ and $q'$ only, in Section 4.1. Then, in Section 4.2, we explain how we designed and trained our efficient verifier $f$, incorporating EF signals. Finally, we explain how such a system can be scaled in Section 4.3.

### 4.1 Deployed Baseline

As a baseline, we consider directly modeling the query variant relation given the two queries as input, as described in Equation 4. To build a verifier, we combine three sources in Section 2 at training/inference:

- Expert Annotation: Training signals can be human-annotated to supervise $f$, though costly and inefficient at scale.
- Retriever and LLM: Retriever can be used as an EF and LLM can be prompted as a verifier.

**Expert Annotation** We obtained 100k expert annotations based on real user queries that have been issued to a commercial search engine. The annotations were obtained from the consensus between two expert annotators, trained and employed at the company, on query pairs with named entities replaced with type tags, essentially yielding a template. Classifying this template ensures that annotations are not biased by the annotators' familiarity with specific entities, and also allows to easily scale 3,725 entity-typed template annotations into a larger dataset consisting of 100k examples by replacing the tag with real-life entities.

We obtained a balanced 1:1 mix of positive and negative annotations, with the negative annotations including hard negatives that have high lexical overlap. Meanwhile, we also consider a public dataset scenario without expert annotations, to show our framework generalizes to diverse scenarios.

**EF on QQ Similarity** An encoder $h$ from the retriever projects both queries into the same latent space, and the resulting similarity score can be directly interpreted as the output of the query variant classification task,

$$s_{qq} = \mathrm{sim}(h(q), h(q')), \tag{6}$$

where $h(\cdot)$ is the embedding function defined by the encoder and $\mathrm{sim}$ is any similarity metric of choice.

**LLM Verifier** In an alternative scenario where LLM calls can be afforded, LLM can be prompted as a verifier, shown in Equation 5. Stronger LLMs, having been exposed to observing variants in a larger amount of context, show stronger performance (Table 2) at an increased inference cost.

### 4.2 Ours: $f_{\text{EF}}$ using Strong EF

Our distinction is to improve $f$ using strong EF signals, over extended context beyond query pair. Figure 1 describes documents from the retriever, which yields an updated verifier $f_{\text{EF}}$ to leverage stronger signals including QD and DD relations:

$$\hat{y} = f_{\text{EF}}(q, q', D = R(q), D' = R(q')). \tag{7}$$

**Encoder** Resembling the baseline architecture, our $f_{\text{EF}}$ also builds on an encoder $h$ extracting top-$k$ retrieved documents from the retriever as EF, and mapping them to features. To this end, $h$ considers queries and retrieved documents simultaneously, and maps them to embedding vectors in a shared latent space and computes the similarity scores between them. In particular, we consider the following closeness features to model the environment feedback:

- QD similarity: The similarity between each of the query and its retrieved document, along with the cross-similarity between the query and the counterpart's documents.
- DD similarity: The pairwise document similarity from the two retrieved sets.

Formally, QD similarity scores are defined as

$$\begin{aligned} s_{qD} &= \big(\,\mathrm{sim}(h(q), h(R_q[i]))\,\big)_{1\leqslant i\leqslant k} \\ s_{q'D'} &= \big(\,\mathrm{sim}(h(q'), h(R_{q'}[i]))\,\big)_{1\leqslant i\leqslant k} \\ s_{qD'} &= \big(\,\mathrm{sim}(h(q), h(R_{q'}[i]))\,\big)_{1\leqslant i\leqslant k} \\ s_{q'D} &= \big(\,\mathrm{sim}(h(q'), h(R_q[i]))\,\big)_{1\leqslant i\leqslant k}, \end{aligned} \tag{8}$$

where $R_q[i]$ denotes the $i$-th ranked document retrieved for query $q$.

Similarities between the query and its retrieved documents, $s_{qD}$ and $s_{q'D'}$, implicitly capture the reliability of the retrieval result for each query. The cross-similarity scores, $s_{qD'}$ and $s_{q'D}$ function as a proxy to measures such as co-click statistics, and also directly model to what extent the retrieval results for the two queries are interchangeable.

DD similarity scores, given as

$$s_{DD} = \big(\,\mathrm{sim}(h(R_q[i]), h(R_{q'}[j]))\,\big)_{i,j}, \tag{9}$$

capture retrieval consistency, modeling how close $R_q$ and $R_{q'}$ are, which serves as an extended PRF. These features augment the model's understanding of equivalence beyond direct query comparisons,

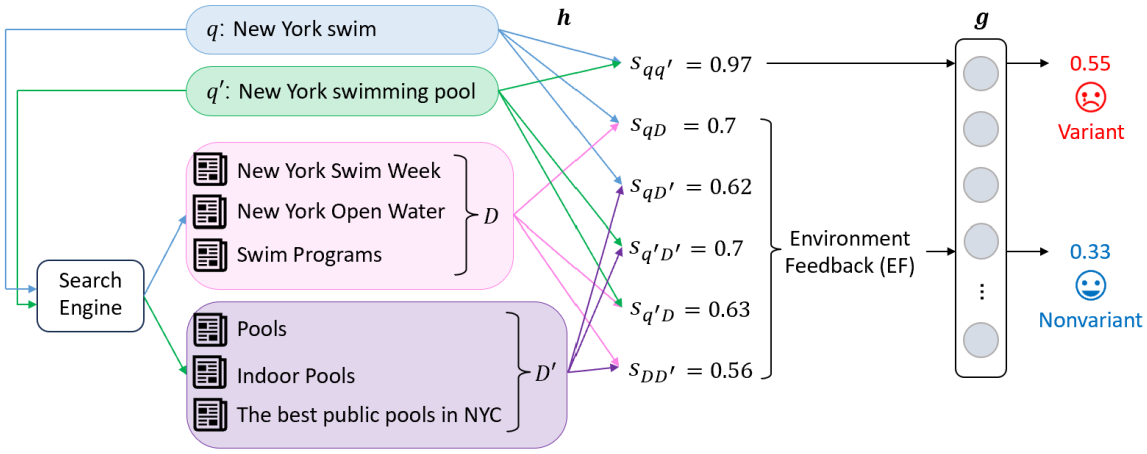


Figure 1: Overall structure of our verifier, which incorporates qq, qd and dd similarities as environment feedback from the retriever to make more informed decisions on query variant identification. For illustration brevity, we show average in place of raw QD and DD similarity scores.

encoded as the QQ similarity score defined in Equation 6.

In total, considering the top-$k$ retrieval results for both queries yields 1 QQ score, $4k$ QD scores, and $k^2$ DD scores for each input pair. Figure 1 illustrates how these scores are obtained and how they contribute to variant detection, though only the average numbers are shown due to presentation brevity. We provide more detailed description of the example in the figure in Section 5.5.

**Predictor** The predictor $g$, an MLP classification head taking the aforementioned similarity scores as input features, aggregates them into a single scalar score which models the probability the given two queries are query variants or not:

$$P(q \sim q') = \sigma(g(s_{qq}, s_{qD}, \cdots, s_{q'D'}, s_{DD})), \tag{10}$$

where $\sigma$ denotes the sigmoid function, i.e., $\sigma(x) = \frac{1}{1+\exp(-x)}$ which maps any real-valued number to a value lying in $(0, 1)$.

**Train Objective and Inference** The predictor $g$ is trained to minimize the binary cross-entropy loss against the ground-truth label $y$, while the encoder $h$ is frozen.

At test time, the predicted probability from the model is converted to a binary classification result with hard thresholding as follows:

$$\hat{y} = \mathbb{1}(P(q \sim q') \geqslant 0.5). \tag{11}$$

**Test-time LLM Prompting with EF** When an LLM call can be afforded, we can inject similarity scores (and their statistics) from the retriever to LLM inference. While scores can be directly passed, providing LLMs with ranked retrieval results in text format, where each document is summarized into a snippet, was more effective: This approach better leverages the LLM's pretrained knowledge to generate more accurate predictions by helping it retrieve and aggregate relevant information from the context. The prompt templates are provided in Appendix A.

### 4.3 Scaling EF

This section discusses how we scale the training dataset (Section 4.3.1) or test-time inference (Section 4.3.2) for improving classification.

#### 4.3.1 Scaling Training Data with Automated Annotation

To avoid reliance on costly expert annotation and efficiently scale training, we utilized the following features to obtain an automatically annotated train set.

**Co-click URLs** Post-search behaviors can function as a strong indicator for query equivalence (Zhang and Dong, 2002; Farzana et al., 2023), as we reviewed in Section 2. Query pairs that co-click URLs above the threshold[2] were considered positive.

**QQ Similarity from LM** As clicks are collected only for exposed documents, and those ranked higher are more likely to be clicked by users (presentation bias), we employed MonoT5 (Nogueira et al., 2020) to compute QQ similarity score as

[2]Empirically set as 100/week.

an additional signal for pairs with fewer co-clicks. This allowed us to mine positive pairs or hard negatives with high MonoT5 similarity.[3] As MonoT5 was trained to model the relevance between a query and a passage/document, $q'$ was fed to the model as if it was a passage associated to $q$.

**Rule-based Rewriting** Expert-written rules, such as swapping or replacing entities, were used to obtain positive pairs by transforming an existing query $q$ to $q'$.

#### 4.3.2 Scaling Test-time Compute with LLM

Our lightweight classifier can be scaled along test-time compute, by predicting in conjunction with an LLM. If the predictions from the LLM and our EF-aware verifier $f_{\text{EF}}$ do not agree,

$$\texttt{LLM}(q, q'; \theta) \neq f_{\text{EF}}(q, q', D, D'), \qquad (12)$$

or in other words, LLM prediction fails the *verification*, a fallback logic is used to determine the output again. As the simplest instantiation of this strategy, we considered invoking a stronger LLM, combining the complementary viewpoint of $f_{\text{EF}}$ and LLM, capturing retriever and pretrained knowledge, respectively.

## 5 Results

### 5.1 Experimental Settings

#### 5.1.1 Benchmarks

We evaluate our method on both proprietary dataset with manual annotation described in Section 4, and also on a public dataset.

**Proprietary Test Set** Proprietary annotation in Section 4.1, was randomly split into training and test sets, each consisting of 50k samples while maintaining a 1:1 ratio of positive to negative samples in both splits.

**Public: PAWS-QQP** We also evaluate our method on a publicly available dataset. Unlike the proprietary set, where features like co-click data can be used to assert that negative pairs are reasonably non-trivial, such signals cannot be collected with public datasets in general.

Specifically, we use the PAWS-QQP (Zhang et al., 2019) benchmark, where all the query pairs are carefully constructed to exhibit high lexical similarity. Stemming from the original QQP (Quora Question Pairs), PAWS-QQP constructed a more challenging set of paraphrase and non-paraphrase pairs by controlling word swaps, applying back translation and evaluating fluency and correctness by human annotators.

As PAWS-QQP only provides the pair of queries $(q, q')$, we used Google cloud custom search engine API to retrieve 10 documents for each query from the web. Then, the document text was obtained by crawling the content of the retrieved URL, followed by processing with `trafilatura`. In addition, as queries in PAWS-QQP have complex sentence forms and tend to span several tens of words in length, we employed GPT-4 to rewrite the queries to mimic real queries issued to search engines, which are typically much simpler. The prompt template used for this query rewriting phase can be found in Appendix A.

#### 5.1.2 Implementation Details and Evaluation Metrics

While our method is orthogonal to the specific choice of encoder and predictor module, we report results with SBERT (Reimers, 2019) used as the encoder $h$. For the classification head $g$, we used a stack of 12 linear layers with output dimension 1 (single scalar output).

The predictor $g$ is trained to minimize the binary cross-entropy loss against the ground-truth label $y$:

$$\mathcal{L}_{\text{BCE}} = -\big(y \log P(q \sim q') + (1-y) \log\big(1 - P(q \sim q')\big)\big). \qquad (13)$$

The encoder $h$ was frozen. We instantiated $g$ as a stack of 12 linear layers with output dimension 1, returning a single scalar output. We used Adam optimizer (Diederik, 2014) with learning rate of 1e-4, weight decay of 1e-4, and the StepLR scheduler with step size of 10 and gamma of 0.5. We trained the model for 100 epochs with an effective batch size of 2048. The experiment was conducted in the environment of Python 3.8.8.

For the LLM, we experimented with two variants from the OpenAI GPT-4 family, namely `gpt-4o-mini` and `gpt-4o`.

For evaluation, we considered two widely used metrics for binary classification tasks, accuracy and F1 score where precision and recall are computed with respect to positive-labeled examples.

### 5.2 Experimental Results

This section validates EF scaling in training and test, as discussed in Section 4.

[3] Empirically tuned with 3+ coclicks and 0.9+ similarity for positive and no coclick and 0.5+ similarity for hard negative.

| | | QQ | | QQ+QD | | QQ+QD+DD | |
|---|---|---|---|---|---|---|---|
| Train | Test | Acc | F1 | Acc | F1 | Acc | F1 |
| Manual | Manual | 80.33 | 80.58 | 82.78 | 81.64 | **83.78** | **84.54** |
| Automatic | Manual | 75.23 | 78.77 | 75.74 | 78.91 | **83.08** | **84.34** |

Table 1: Accuracy and F1 scores of our verifier, trained with manual and automatic train set evaluated on manual test set from proprietary dataset. Best results are boldfaced, demonstrating the effectiveness of EF.

| | | Proprietary | | PAWS-QQP | |
|---|---|---|---|---|---|
| Method | Verifier | Acc | F1 | Acc | F1 |
| *Reference: LLM classifiers* | | | | | |
| LLM-only (GPT-4o mini) | – | 86.84 | 86.72 | 64.52 | 56.18 |
| LLM-only (GPT-4o) | – | <u>88.14</u> | <u>87.83</u> | **68.76** | **58.74** |
| *Ours* | | | | | |
| Ours (lightweight) | – | 83.78 | 84.54 | 65.53 | 56.53 |
| LLM (GPT-4o mini) + Verification with Ours | $f_{\text{EF}}$ | **88.65** | **88.56** | – | – |
| " | weak EF | – | – | <u>66.55</u> | <u>57.82</u> |

Table 2: Results on proprietary and public (PAWS-QQP) test sets. Best results are boldfaced, while the second best is underlined, without consideration of costs.

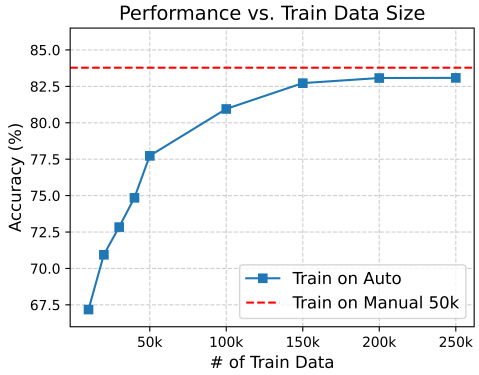


Figure 2: Accuracy versus train data size shows auto train data can lead to comparable performance to manual, when scaled to 5-fold in size.

First, Table 1 shows scaling input features to accommodate more diverse EF during training, such as QD and DD similarities, yields performance gains. Notably, these gains are more significant when $f$ is trained on automatically collected data. A qualitative example illustrating how EF informs predictions is provided in Section 5.5.

Second, Figure 2 highlights that increasing the size of the training data improves performance. Using only auto-labeled data, the model achieves results comparable to those obtained with a manual training set.

Finally, Table 2 illustrates the benefits of scaling test-time compute by integrating LLMs into our framework, which we dive deeper with two research questions **RQ1** and **RQ2**.

### 5.3 RQ1: Integrating LLM with Ours

The lightweight $f_{\text{EF}}$, trained on a proprietary dataset and optimized for latency-sensitive scenarios, naturally underperforms, when unfairly compared to standalone LLM classifiers designed for higher computational budget.

In this new high budget scenario, we show EF signals from $f_{\text{EF}}$ combines with predictions from a smaller LLM, GPT-4o mini, to achieve higher accuracy than a larger LLM alone (as shown in the 4th row of Table 2).

Moreover, selectively delegating to the larger LLM only when the verifier disagrees with the smaller LLM's prediction reduces calls to the larger model to less than 20%, while still improving performance. This demonstrates that when the pretrained knowledge of the LLM aligns with explicit EF signals from the search engine, the result is more reliable than relying solely on a more powerful model like GPT-4o.

### 5.4 RQ2: Generalization to Public Data

For the PAWS-QQP dataset, EF from retriever is limited solely to retrieved documents, or "weaker EF" than Proprietary dataset, where additional features like co-clicks or expert annotations are provided.

Our findings on this public benchmark, denoted as weak EF in Table 2, are as follows:

- Even with weaker EF, performance improves compared to the LLM-only baseline.
- However, weaker EF does not surpass the stronger LLM, while stronger EF does so.

### 5.5 Qualitative Example

Finally, in order to qualitatively illustrate how EF guides the prediction, we consider Figure 1 as a running example. Given the query pair ($q$: "New York swim", $q'$: "New York swimming pool"), predicting solely based on QQ similarity would lead to a false positive, as $q$ and $q'$ are lexically similar.

However, their search intents are distinguished clearly: $q$ is likely a general search related to swimming, such as swimming competitions, swimming programs for lessons, swimsuits or beachwear, or Swim Week, a fashion week for swimwear. In comparison, $q'$ is more specific to swimming pool locations, facilities, or contact information.

Such discrepancy can be detected from EF features, especially $s_{qD'}$, scoring lower than the global average similarity scores for negative pairs strongly indicate non-equivalence. While the actual design of $f_{\text{EF}}$ leverages individual similarity scores to support signals in diverse granularity, we simplified to show the average scores for illustration brevity; still, it is captured in the average similarity scores as well that the search results for this example are not so interchangeable and that the retrieved documents exhibit notably low similarity in general, a strong indicator for non-variant pairs.

## 6 Conclusion

In this paper, we explored the use of EF to identify query variants. Our findings demonstrate that our approach substantially outperforms deployed baselines, in both budget-constrained and less restricted scenarios. In addition, we release the expert annotations to support future development in this area.

## Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00414981), and by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00077/RS-2022-II220077, AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data).

## References

Elham Azhir, Nima Jafari Navimipour, Mehdi Hosseinzadeh, Arash Sharifi, and Aso Darwesh. 2021. An automatic clustering technique for query plan recommendation. *Information Sciences*, 545:620–632.

Doug Beeferman and Adam Berger. 2000. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416.

Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883.

Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R Bowman, and Kyunghyun Cho. 2023. Two failures of self-consistency in the multi-step reasoning of llms. *arXiv preprint arXiv:2305.14279*.

I Chien, Chao Pan, and Olgica Milenkovic. 2018. Query k-means clustering and the double dixie cup problem. *Advances in Neural Information Processing Systems*, 31.

P Kingma Diederik. 2014. Adam: A method for stochastic optimization. *(No Title)*.

Shahla Farzana, Qunzhi Zhou, and Petar Ristoski. 2023. Knowledge graph-enhanced neural query rewriting. In *Companion Proceedings of the ACM Web Conference 2023*, pages 911–919.

Jianfeng Gao, Chris Quirk, et al. 2010. A large scale ranker-based system for search query spelling correction. In *The 23rd international conference on computational linguistics*.

Hiroki Iida and Naoaki Okazaki. 2021. Incorporating semantic textual similarity and lexical matching for information retrieval. In *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation*, pages 582–591.

Victor Lavrenko and W Bruce Croft. 2003. Relevance models in information retrieval. In *Language modeling for information retrieval*, pages 11–56. Springer.

Mu Li, Muhua Zhu, Yang Zhang, and Ming Zhou. 2006. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1025–1032.

Sen Li, Fuyu Lv, Taiwei Jin, Guiyang Li, Yukun Zheng, Tao Zhuang, Qingwen Liu, Xiaoyi Zeng, James Kwok, and Qianli Ma. 2022. Query rewriting in taobao search. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3262–3271.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Rodrigo Paredes and Edgar Chávez. 2005. Using the k-nearest neighbor graph for proximity searching in metric spaces. In *String Processing and Information Retrieval: 12th International Conference, SPIRE 2005, Buenos Aires, Argentina, November 2-4, 2005. Proceedings 12*, pages 127–138. Springer.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Krishna Srinivasan, Karthik Raman, Anupam Samanta, Lingrui Liao, Luca Bertelli, and Michael Bendersky. 2022. QUILL: Query intent with large language models using retrieval augmentation and multi-stage distillation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 492–501, Abu Dhabi, UAE. Association for Computational Linguistics.

Fei Wang, Chao Shang, Sarthak Jain, Shuai Wang, Qiang Ning, Bonan Min, Vittorio Castelli, Yassine Benajiba, and Dan Roth. 2024. From instructions to constraints: Language model alignment with automatic constraint verification. *arXiv preprint arXiv:2403.06326*.

Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2001. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*, pages 162–168.

Dell Zhang and Yisheng Dong. 2002. A novel web usage mining approach for search engines. *Computer Networks*, 39(3):303–310.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. 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 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Prompt for Query Rewriting for Benchmark Preprocessing

Given a question in its natural sentence form, convert it into a more concise format that is more likely to be issued as a search query to search engines. The search intent of the user must be preserved. As in the following examples, decide whether the two given queries are equivalent or not.
Here is the question in sentence form, convert it to concise form that is more likely to be a real search query.
**Question**: {`query` ($q$)}
**Answer**:

Figure 3: Prompt for rewriting the query in PAWS-QQP.

Prompt for Classifying Query Variant without EF

The equivalent query condition requires that both queries have the same search intent, and that if the same search result is presented to the user for both queries, the user's satisfaction level should be the same as well. As in the following examples, decide whether the two given queries are equivalent or not. Your final answer should be either 'Yes' or 'No'.
Here are the two queries to be tested for equivalence:
**Query 1**: {`query 1` ($q$)}
**Query 2**: {`query 2` ($q'$)}
**Answer**:

Figure 4: Prompt for deciding query equivalence.

## A Prompt Template Examples

Here we provide prompt templates used for inference with LLMs. Figure 3 shows the prompt used for rewriting the queries in the PAWS-QQP benchmark to follow more realistic styles, Figure 4 shows the prompt for deciding query equivalence, and Figure 5 shows the prompt for incorporating environment feedback through prompting.

Prompt for Classifying Query Variant with EF

The equivalent query condition requires that both queries have the same search intent, and that if the same search result is presented to the user for both queries, the user's satisfaction level should be the same as well. As in the following examples, decide whether the two given queries are equivalent or not. Your final answer should be either 'Yes' or 'No'.
In addition to the queries themselves, you will be also provided with top-10 search results from the search engine, with titles and summarized snippets from each retrieved web document. Analyze the similarities and dissimilarites in search results to make your decision more informed. But remember, search engines can also fail, giving results with lots of discrepancies even if the real user intent was staying the same, or vice versa. And more importantly, the rankings themselves encode lots of information as well.
Here are the two queries to be tested for equivalence:
**Query 1**: {`query 1` ($q$)}
**Query 2**: {`query 2` ($q'$)}
And here is the search result summarization:
[Search result for Query 1]
Title: {`title of document 1 for query 1`}
Snippet: {`summarization of document 1 for query 1`}
...
Title: {`title of document 10 for query 1`}
Snippet: {`summarization of document 10 for query 1`}
[Search result for Query 2]
...
Title: {`title of document 10 for query 2`}
Snippet: {`summarization of document 10 for query 2`}
But remember, your goal is to decide if the following two queries have the same search intent or not, think about whether the user's satisfaction would be the same even if the search results are exchanged. These search results were not tested on the user who issued these queries, and it is not known whether these results are satisfactory or not.
**Query 1**: {`query 1` ($q$)}
**Query 2**: {`query 2` ($q'$)}
**Answer**:

Figure 5: Prompt for deciding query equivalence with environment feedback.