Comparative Study of Query Types on Query Refinement

Zahra Taheri[1], Ziad Kobti[1] and Hossein Fani[1*]

[1]School of Computer Science, University of Windsor, Windsor, ON, Canada.

*Corresponding author(s). E-mail(s): hfani@uwindsor.ca;
Contributing authors: taherik@uwindsor.ca; kobti@uwindsor.ca;

Abstract

Within a search session, users seek their information needs through iterative refinement of their queries, which is daunting. Neural-based query refinement methods aim to address this challenge via training on gold-standard pairs of ⟨original query → *refined* query⟩, oblivious to the type of queries; be it `factual` like *'table leafs or leaves'*, `navigational`, e.g., *'bbc good food recia'*, or `transactional` such as *'free play minecraft'* and, hence, fall short of finding refined versions for many original queries. In this paper, we bridge the gap by incorporating query types when generating refined queries. We fine-tune a conditional transformer, e.g., `t5`, to map an original query onto its relevant documents while conditioning on its type during training so that, during the inference, the query type controls generating new reformulated queries within the same search intent for an *unseen* original query. Our experiments on the large-scale `orcas-i-2m` dataset across five query types demonstrated the synergistic effects of considering query types in generating more refined queries with better information retrieval efficacy. Specifically, considering query type has shown the best retrieval performance among `navigational` queries while yielding the highest number of refined queries in `factual` queries. The codebase to support the reproducibility of our research is available at https://github.com/fani-lab/RePair/tree/type-aware

Keywords: Query Type, Search Intent Taxonomy, Query Suggestion, Query Reformulation, Conditional Transformer

Table 1: Sample queries from `orcas-i-2m` [4] and the search efficacy of type-aware vs. type-less refinements using `t5` [47]. As *underlined*, using query types yields better refinements.

	qid	type	query	bm25.ap (Δ)
original	2009263	–	*bbc good food recia*	0.039
−type	2009263	–	*bbc good foods recipies*	0.266 (+0.277)
+ type	2009263	navigational	*bbc good food http://www.bbcgoodfood.com/recipes*	**0.566** (+0.577)
original	2016037	–	*table leafs or leaves*	0.111
−type	2016037	–	*table leaf leafs*	0.100 (−0.011)
+ type	2016037	factual	*table leaf definition*	**1.000** (+0.889)
original	11136851	–	*free play minecraft*	0.041
−type	11136851	–	*minecraft full version*	0.066 (+0.025)
+ type	11136851	transactional	*minecraft mineroll download free game*	**1.000** (+0.959)

1 Introduction

Users of search engines struggle with formulating queries to express their information needs due to an unclear/uncertain state of knowledge [6, 7, 31], inconsistent terminologies (vocabulary mismatch) [24, 34, 53, 64], or pure lack of knowledge, which may result in less-than-relevant search engine result pages. Query refinement aims to bridge the gap by recommending a *refined* query for the user's original query, which retrieves more relevant information at higher ranks [3, 5, 40, 48, 53, 57, 65]. Query refinement methods have been effective in web search [66], e commerce [26], and medical information retrieval [2, 8]. However, methods of query refinement largely overlook the type of user's query, and hence, fail to find refined queries for many original queries. For example, as shown in Table 1, the original query *'table leafs or leaves'* is poorly refined to *'table leaf leafs'*, decreasing the average precision for `bm25` retriever from `0.111` down to `0.100 (−0.011)` as the query type has been overlooked. In contrast, knowing the query type as `factual` can specify user expectations and influence refinement such that a type-aware model generates a better refined query *'table leaf definition'*, effectively detecting the user's information need and enhancing average precision to its best, i.e., `1.0 (+0.889)`. For example, as shown in Table 1, the original query *'free play minecraft'* is poorly refined to *'minecraft full version'*, marginally increasing the `map` from `0.041` to `0.066` (+`0.025`) as the query type has been overlooked (−type). In contrast, knowing the query type as `transactional`, a type-aware model generates a better refined query *'minecraft mineroll download free game'*, effectively detecting the user's information need and enhancing `map` to its best, i.e., `1.0` (+`0.959`).

Query types, also known as search intent taxonomies, have long been integrated into information retrieval processes to enhance search efficacy [9, 16, 18, 29, 51, 52]. For web searches, Border [10] identifies three types of queries: `informational`, like *'table leafs or leaves'*, aiming at gaining specific information assumed to be present on one or more webpages, `navigational`, like *'bbc good food recipies'* to find a specific url or website, and `transactional`, like *'minecraft mineroll download free game'*, to access a webpage inside which further interaction would occur. Choi et al. [18] studied *'how to'* queries about

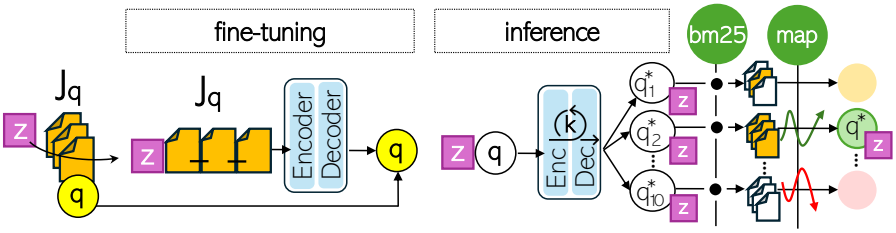

Fig. 1: Utilizing conditional transformers for type-aware query refinement.

performing procedural tasks and identified query types based on the underlying tasks like creating/building vs. fixing. Researchers have also proposed the automatic generation of query type taxonomies from search logs and the classification of queries into these types accordingly [1, 17, 19, 52]. Yet, the synergistic effect of considering queries' types for query refinement has remained *un*explored within the large body of query refinement methods [3, 5, 40, 48, 57], including the state of the art where rich contextual information, such as past queries and click-throughs, are considered [3, 65]. Chen et al. [41] integrated users' short-term and long-term search histories within their current session to identify search interests. Ahmad et al. [3] encoded users' previous queries and document clicks in the search log into the recurrent neural network model. Later, Zhou et al. [65] divide users' search logs into long and short-term histories to refine the initial query using the transformer model. Fiorini et al. [42] fused personalization and time-sensitive information to investigate the difference in query occurrences throughout a day. For example, the query *'tv guide'* appears mainly in the evenings and nights. Therefore, they incorporated the time of the issued query along with users' previous queries in their proposed model. Recently, researchers argued that web query logs are prone to topic drift [5, 40, 57], that is, users' intent may undergo gradual or sudden changes within a search session, resulting in a loss of sequential semantic context between queries, and proposed generating gold-standard benchmark datasets free of topic drift for supervised refinement methods. Nonetheless, existing works in query refinement forego query types in their proposed methods.

In this paper, we propose to study the impact of considering query types and lack thereof for query refinement via a conditional transformer. As shown in Figure 1, given an original query and its query type, we form a training pair ⟨condition: input → output⟩ where the condition is the query type, the input is the concatenation of relevant documents, and the output is the original query. We train `t5` [47] to learn the transformation from relevant documents of the original query to the query itself but on the condition of its type. During inference, we feed an *unseen* original query from the test set with its query type to `t5` to generate different streams of words in the output using top-k sampling [23] as the type-aware refined version of the original query. We evaluate the search efficacy of the type-aware refined version compared to the original query as well as a refined version without considering the query type [5, 38]

using a lightweight retriever such as `bm25` [49] and an evaluation metric such as average precision (`ap`). Our experiments demonstrate that incorporating query types for refining queries through `t5` statistically significantly enhances the efficacy in retrieving more relevant information yet at higher ranks and, hence, yields better refined queries compared to when the query type is overlooked.

2 Related Work

In this section, we provide a concise summary of query refinement methods and the applications of transformers in this context.

2.1 Query Refinement

Earlier approaches of query refinement, variously referred to by such other names as query rewriting, reformulation, or expansion, were mostly unsupervised wherein an original query was modified by expanding and/or replacing words with synonyms [33, 54, 55] or other frequently co-occurring words within a training corpus [36, 39]. Such methods, however, overlooked the query's semantic context for polysemous words and yielded topic drift. To fill the gap, Rocchio [50] and others [28] proposed to modify the query based on words in the set of clicked documents as relevance feedback from users or via pseudo-relevance-feedback [11, 12, 58, 60]. Successful for short queries, unsupervised methods fall short for detailed and long queries. Hence, semi-supervised and supervised techniques, mostly based on neural recurrent encoder-decoder architecture, were proposed to generate a refined query [3, 13, 20, 22, 25, 35, 56, 61, 63]. For instance, Ahmed et al. [3] incorporate historical ⟨query, clicked documents⟩ pairs to learn multitask of query refinement and document ranking in tandem. Supervised and semi-supervised methods, by and large, are trained on search logs, assuming that a user would gradually refine her query over successive attempts to find relevant content within a search session, which has been *in*validated by Chen et al. [15]; a user might search for multiple topics in one session, and hence, *ir*relevant queries would be paired. Hence, standard benchmark datasets free of topic drift are specifically designed for supervised query refiners [5, 57, 62]. For instance, Arabzadeh et al. [5] fed a query to the pretrained `doc-t5-query` [43] and selected the generated sequence of words as a refined version only if it increases `bm25` retrieval performance based on `map`. Also, Narayanan et al. [40] developed an open-source reproducible pipeline to generate benchmark datasets from various domains and any choice of transformers. In the healthcare domain, Rencis [67] applied a configurable keywords-based query language to address the challenges of domain-specific retrieval, demonstrating how structured query refinement tailored to specialized vocabularies can enhance retrieval precision in clinical information systems. Nonetheless, to the best of our knowledge, no one has yet explored the synergistic impact of query types in query refinement.

2.2 Conditional Transformers

Transformers have been widely employed in query refinement. [5, 27, 44, 59]. To address vocabulary mismatch between the original queries and relevant documents, Nogueira et al.[44] proposed `doc2query` to expand documents by semantically related words using the transformer [59]. Shortly after, they replaced the transformer with `t5` [47], observing a substantial improvement in retrieval efficacy. Arabzadeh et al. [5] used `t5` to generate training datasets for supervised query refinement at a large scale. Also, Mitra et al. [38] utilized `bert` [21] to generate related queries to an original query (*'people also asked'*). Recently, Hosseini et al. [27] used `doc-t5-query` to refine queries and trained a classifier to filter out refinements that semantically drift from the original query.

The promising performance of transformers paved the way for *conditional* transformers, where probabilities are assigned to sequences of words given a pretext that encodes a condition. Formally, $p(q \mid \texttt{X}) = \prod_i p(w_{i+1} \mid \texttt{X}, w_1, \ldots, w_i)$, where the probability of the next word is based on previously generated words and a conditioning context `X`, which is kept fixed during the encoding and decoding phases. For instance, Keskar et al. [30] proposed `CTRL`, a transformer that can condition on a predefined set of words to influence the *genre* of the generated sequence of words in the output for the same input sequence; for example:

$$p(\textit{sharp} \mid \texttt{X} = \textit{review}, \textit{knife}) \quad \text{vs.} \quad p(\textit{tool} \mid \texttt{X} = \textit{wiki}, \textit{knife})$$

Subsequently, conditional transformers have been employed as single universal (generalist) models, such as `T5`, to perform a wide range of natural language processing tasks—from question answering to document summarization and sentiment analysis—relieving the need for task-specific models [30, 37, 46]. Inspired by `T5` and `CTRL`, we aim to leverage conditional transformers to refine original queries conditioned on their types (with `X` representing the query type), in order to study the impact of query types on query refinement.

3 Problem Definition

Let q represent a query and $\mathcal{J}_q$ be q's relevant documents (relevance judgment). Given an evaluation metric m to evaluate the quality of retrieved documents D_{rq} by a retriever r for q regarding $\mathcal{J}_q$, denoted as $m(D_{rq} : \mathcal{J}_q)$, a query q^* is a *refined* query for q iff $m(D_{rq^*} : \mathcal{J}_q) > m(D_{rq} : \mathcal{J}_q)$. Type-aware query refinement aims to generate a better refined query q^*_z owing to q's type z such that:

$$m(D_{rq^*_z} : \mathcal{J}_q) > m(D_{rq^*} : \mathcal{J}_q) > m(D_{rq} : \mathcal{J}_q) \tag{1}$$

For instance, from Table 1, the type-aware refined query q^*_z=*'minecraft mineroll download free game'* enables the retriever r=`bm25` to retrieve *all* relevant documents $D_{rq^*_z}$ and obtain the evaluation metric m=`map` at its best `1.00` compared to the documents retrieved by the type-less refined query

q^*='*minecraft full version*' and the original query q='*free play minecraft*' with m=`map` equal to `0.066` and `0.041`, respectively.

4 Research Methodology

Let τ denote a conditional transformer, such as `T5` [47]. In type-aware query refinement, we define the condition `X` as the query type z, and aim to train τ to perform conditional translation from the words in relevant documents $\mathcal{J}_q$ to the words in the original query q. That is, we aim to model the distribution of query words given their relevant documents and their type: $p(q \mid \texttt{X} = z, \mathcal{J}_q)$, where z controls the refinement process.

During training, given the original query q and its relevant documents $\mathcal{J}_q$, we concatenate the documents into a single input sequence, prepend the query type z as a prefix condition (e.g., z: $\mathcal{J}_q$), and pair it with the original query as the target sequence, forming an input-output training pair: *(z:* $\mathcal{J}_q$*)* $\rightarrow$ *q*. The model τ is thus expected to learn different refinement pathways depending on the query type.

At inference time, the model τ is provided with a test query q along with its type z, formatted as *(z: q)*. The transformer generates a refined version of q conditioned on z, denoted q^*_z, by producing the mapping: *(z: q)* $\rightarrow$ q^*_z.

Modern transformers, including conditional ones, apply top-k random selection at their decoders by randomly selecting the next word of the output stream from the top-k most probable words to generate a varied stream of words and avoid common or repetitive words from training datasets [23, 44]. Top-k random selection, hence, yields non-deterministic output generation during inference given the same input. To ensure diverse and novel refined versions of the same original query, we employ top-k random selection, i.e., $\langle \langle z : q \rangle \rightarrow q^*_{zi} \rangle$ for $1 \leq i \leq k$, as opposed to a single refined query. To compare type-aware query refinement with when query types are overlooked, we follow similar steps for the same set of queries but oblivious to their query type with no pretext `X` in τ.

We argue that type-aware query refinement results in higher search efficacy and yields better and a greater number of refined queries q^*_z compared to q^* where query types are overlooked.

We create two sets of pairs $\mathcal{Q}^z = \{\langle z : \mathcal{J}_q \rightarrow q \rangle\}$ for the type-aware method, and lack thereof $\mathcal{Q} = \{\langle \mathcal{J}_q \rightarrow q \rangle\}$ for the type-less baseline [5, 38]. We then fine-tune τ on $\mathcal{Q}^z$ (type-aware model) and $\mathcal{Q}$ (type-less model) separately and collected the models' inference results for the test set, which results in two categories of type-aware and type-less query refinements. Finally, we assessed the effectiveness of refined queries in each category by evaluating how well they perform in retrieving the relevant documents based on the original query's relevance judgment $\mathcal{J}_q$. Given a query q from the test set, we pick its type-aware refined versions q^*_{zi} to retrieve documents $D_{rq^*_{zi}}$ using a retriever r, and evaluate the query's performance using an information retrieval metric m, i.e., $m(D_{rq^*_{zi}} : \mathcal{J}_q)$. For the same query q, we also pick a type-less refined version

Table 2: Statistics of `orcas-i-2m` [4]. $|q|$ and $|\mathcal{J}_q|$ denote the number of words and relevant documents for a query q.

Type	Total Queries ($\|\mathcal{Q}\|$)	Train	Test	Avg. $\|q\|$	Avg. $\|\mathcal{J}_q\|$
All	2,000,000	1,437,272 (80%)	359,402 (20%)	3.438	2.833
`navigational` *(specific URL or resource)*	289,610 (14.4%)	210,175 (14.6%)	52,561 (14.6%)	2.257	2.054
`factual` *(knowledge or facts)*	706,907 (35.3%)	511,284 (35.5%)	127,666 (35.6%)	3.493	3.169
`transactional` *(resources or downloads)*	83,334 (4.1%)	60,028 (4.1%)	15,035 (4.2%)	3.920	3.999
`instrumental` *(task-oriented knowledge)*	116,485 (5.8%)	83,755 (5.8%)	20,939 (5.8%)	4.344	3.155
`abstain` *(none of the above)*	803,664 (40.8%)	572,017 (39.7%)	143,193 (39.8%)	3.168	1.774

q_i^*, where the query type z was overlooked, and use it to retrieve documents $D_{rq_i^*}$. We argue that, on average, E.q. 1 holds true for a randomly selected query q from the test set across different retrievers r and evaluation metrics m, and hence, considering the queries' types through type-aware query refinement results in higher search efficacy and yields better refined queries compared to when query types are overlooked. Furthermore, with both type-aware and type-less models at our disposal, we posit that the type-aware model performs on par with or even surpasses the type-less model for queries with unknown types, a scenario more reflective of real-world applications. It is worth noting that detecting a type for a query is *not* the focus of this paper. Herein, our main goal is to study the synergistic effects of considering the query's type, known *a priori*, for its refinement. Extending type-aware query refinement for queries whose type is missing or unknown by an end-to-end query type detection *and* refinement is indeed our future research direction.

5 Experiment

We seek to answer the following research questions:
RQ1: Does considering query types enhance the performance of refined queries?
RQ2: Is the type-aware query refinement consistently effective across all query types?
RQ3: Is the efficacy of the type-aware query refinement consistent across different retrievers?
RQ4: How does the type-aware model perform compared to its type-less counterpart when the query type is *un*known?

5.1 Setup

5.1.1 Dataset.

To our knowledge, only two publicly available benchmark query sets classify queries by type; `trec-mq-2009` [14], which contains `420` queries across `6` types but it is small and insufficient for fine-tuning a transformer. Therefore, we used the large-scale `orcas-i-2m` [4], which comprises `2 million` queries of `5`

Table 3: Results of type-aware vs. type-less query refinement on the test set. † refers to statistically significant improvements using one-tailed (greater) paired `t-test` with `99%` confidence interval (p-value $<$ `0.01`).

	bm25						qld					
	map		ndcg		mrr		map		ndcg		mrr	
	avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$
original	0.174	–	0.284	–	0.253	–	0.170	–	0.281	–	0.248	–
type-less	0.279	272,936	0.420	272,936	0.416	272,909	0.272	275,734	0.406	275,730	0.415	275,727
type-aware	**0.289**†	**287,396**	**0.446**†	**287,387**	**0.451**†	**287,369**	**0.282**†	**281,607**	**0.412**†	**281,602**	**0.426**†	**281,607**

Table 4: Results of type-aware vs type-less query refinement *per query type* on the test set. † refers to statistically significant improvements using one-tailed (greater) paired `t-test` with `99%` confidence interval (p-value $<$ `0.01`).

	$\#q$	bm25						qld					
		map		ndcg		mrr		map		ndcg		mrr	
		avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$	avg	$\#q^*$
`navigational`													
original		0.128	–	0.200	–	0.185	–	0.116	–	0.185	–	0.173	–
type-less	52,561	0.305	34,605	0.440	34,605	0.433	34,605	0.286	33,928	0.422	33,928	0.418	33,928
type-aware		**0.325**†	**36,914**	**0.460**†	**36,914**	**0.462**†	**36,914**	**0.302**†	**35,241**	**0.438**†	**35,241**	**0.440**†	**35,241**
`factual`													
original		0.188	–	0.308	–	0.265	–	0.197	–	0.323	–	0.275	–
type-less	127,666	0.311	103,407	0.463	103,407	0.448†	103,396	0.319†	106,195	0.477†	106,195	0.460†	106,195
type-aware		**0.320**†	**111,659**	**0.475** †	**111,659**	**0.463**†	**111,648**	**0.328**†	**111,838**	**0.482**†	**111,838**	**0.467**†	**111,838**
`transactional`													
original		0.136	–	0.230	–	0.209	–	0.121	–	0.212	–	0.189	–
type-less	15,035	0.272	11,276	0.413	11,276	0.424	11,276	0.248	11,072	0.389	11,072	0.396	11,072
type-aware		**0.288**†	**12,175**	**0.430** †	**12,175**	**0.448**†	**12,175**	**0.255**†	**12,175**	**0.397**†	**12,175**	**0.405**†	**12,175**
`instrumental`													
original		0.161	–	0.279	–	0.247	–	0.147	–	0.266	–	0.227	–
type-less	20,939	0.269	16,593	0.422	16,593	0.425	16,586	0.245	16,787	0.400	16,787	0.384	16,780
type-aware		**0.276**†	**17,479**	**0.435** †	**17,479**	**0.434**†	**17,472**	**0.251** †	**17,126**	**0.403**†	**17,126**	**0.395**†	**17,126**
`abstain`													
original		0.133	–	0.296	–	0.269	–	0.171	–	0.284	–	0.257	–
type-less	143,193	0.292	107,055	0.430	107,055	0.450	107,046	0.273	**107,752**	0.315	**107,748**	0.427	**107,752**
type-aware		**0.294**†	**109,169**	**0.431** †	**109,169**	**0.451**†	**109,169**	**0.275**†	105,821	**0.320**†	105,821	**0.428**†	105,821

types: `factual`, `instrumental`, `navigational`, `transactional`, and `abstain` (*un*known).

5.1.2 Conditional Transformer.

We fine-tuned the pretrained `t5-base` [47] with `220 million` parameters for `4,000` epochs on the train set with a batch size of `256` and a learning rate of `0.001`. The maximum sequence lengths for the input and output are set to `2,048` and `32`, respectively. We chose `t5` for its seamless integration of conditions, herein, a query type, into the training and inference phases by prepending it as a textual prefix followed by ':' to the input. For type-less query reformulations, we disregard the query type. Also, `t5`'s straightforward installation process, open-source codebase, and efficient use of google cloud's tpus help speed up model fine-tuning and inference. During inference, we utilized top-k = `10` random selection in the decoder, which resulted in `10` type-aware and `10` type-less refinements for each original query.

5.1.3 Evaluation.

We assess the efficacy of type-aware refined queries in comparison with type-less refined versions when retrieving the relevant documents based on the original query's relevance judgment. We use different lightweight sparse retrievers including `bm25` [49] and `qld` [45], and evaluate the retrieval performance using mean average precision (`map`), mean reciprocal rank (`mrr`), and normalized discounted cumulative gain (`ndcg`) at cutoff `100`, followed by a statistical significant test (†) using one-tailed (greater) paired `t-test` with `99%` confidence interval (p-value $<$ `0.01`). We acknowledge the state-of-the-art retrieval performance of dense retrievers, such as `colbert` [32]. However, given the intensive computational demands such methods impose on our large-scale query set, coupled with our limited computational resources, we were unable to include them in our testbed. Our primary goal, however, is to demonstrate the efficacy of incorporating query types in refining queries; a goal achievable even with lightweight off-the-shelf sparse retrievers. We expect that advanced dense retrievers would likely demonstrate similar effects.

5.2 Results

In response to **RQ1**, i.e., whether considering query types enhances the performance of refined queries, from Table 3, we observe that refined queries by the type-aware model statistically significantly enhance the search experience compared to the refined queries by the type-less model as well as the original queries. The improvements across all evaluation metrics are more evident when employing `qld` as the retriever. While we may interpret the improvements for the evaluation metrics as marginal, by looking at the number of original queries that could be matched with a refined version, we can observe a substantial increase when query types are considered. Specifically, using `bm25`, the type-aware model generates about `14,000` more refined queries compared to the type-less model. In summary, the utilization of query types substantially helped many original queries that would otherwise have been left without refinement.

Regarding **RQ2**, that is, if type-aware query refinement is consistently effective across all query types, Table 4 shows that incorporating query types enhances query refinement for *all* query types consistently and statistically significantly across evaluation metrics and retrieval methods. Notably, integrating query types leads to significant improvements in the refinement of `navigational` queries across all metrics, even though they constitute only 14% of our training set. It is also worth noting that achieving improvement in `factual` and `instrumental` queries was unexpected, as they are often less ambiguous compared to other types, and the related documents for these queries tend to have overlaps. Moreover, despite marginal improvements in metrics, we observed a major increase in the number of generated refined queries by the type-aware model, ranging from `800` for `instrumental` to `8,000`

more refined queries for `factual` queries. In contrast, `abstain` queries demonstrate the smallest improvements in refinement and fall short of the increase in the number of refined queries, esp., when using the `qld` retriever, which is expected as the query type is unknown and offers little help to the model.

To answer **RQ3** about the efficacy of the type-aware query refinement across retrievers, from Tables 3 and 4, we observe that both `bm25` and `qld` retrieve more relevant documents when applied to the type-aware refined queries. These findings indicate that type-aware query refinement enhances retrieval efficacy, irrespective of the information retrieval methods. Additionally, the number of queries matched with a refined version from the type-aware model has consistently improved per query type across all metrics and retrievers, except for the `abstain` queries when `qld` is the retriever.

In response to **RQ4** about the performance of type-aware vs. type-less models for queries with unknown type, looking at Table 4 for `abstain` queries, we observe that the type-aware model performs better than its type-less counterpart. The improvement is more pronounced using `bm25` as the retriever, wherein both the metric values and the number of refined queries are increased. This makes the type-aware model a more suitable choice for real-world applications where query types may *not* be explicitly known.

Our findings align with Arabzadeh et al. [5] and Narayanan et al. [40], who demonstrated that semantic conditioning and query reformulation can improve retrieval precision. However, unlike their type-agnostic approaches, our method conditions refinements explicitly on query type, resulting in larger improvements, particularly for navigational queries. This observation complements Rajaei et al. [48], where semantic drift was reduced through backtranslation, by showing that query-type awareness can further mitigate such drift. Together, these comparisons indicate that query-type awareness offers a complementary advantage to existing refinement techniques and is broadly applicable across domains.

6 Concluding Remarks

In this paper, we studied the impact of query types on query refinement. Using a conditional transformer, we fed query types as a prior condition to generate type-aware refinements of queries. Our experiments on a large-scale query set demonstrated statistically significant improvement across five query types when considering query types versus lack thereof. Per query type, `navigational` queries benefited the most in terms of evaluation metrics, while the `factual` queries obtained a higher number of refined queries. As we obtain the necessary computational resources, our future research includes other conditional transformers as well as dense retrievers to explore whether our findings generalize. Also, we aim to extend type-aware query refinement to an end-to-end in-tandem approach for query type detection and refinement to better suit real-world scenarios where query types may not known in advance.

References

[1] M. Agosti, F. Crivellari, and G. M. Di Nunzio, Web log analysis: a review of a decade of studies about information acquisition, inspection and interpretation of user interaction, Springer, New York, 2012.

[2] M. Agosti, G. M. Di Nunzio, and S. Marchesin, An Analysis of Query Reformulation Techniques for Precision Medicine, ACM, Paris, 2019.

[3] W. U. Ahmad, K.-W. Chang, and H. Wang, Context Attentive Document Ranking and Query Suggestion, ACM, Paris, 2019.

[4] D. Alexander, W. Kusa, and A. P. de Vries, ORCAS-I: Queries Annotated with Intent using Weak Supervision, ACM, Madrid, 2022.

[5] N. Arabzadeh, A. Bigdeli, S. Seyedsalehi, M. Zihayat, and E. Bagheri, Matches Made in Heaven: Toolkit and Large-Scale Datasets for Supervised Query Reformulation, ACM, Queensland, 2021.

[6] L. Azzopardi and J. Liu, Search under Uncertainty: Cognitive Biases and Heuristics - Tutorial on Modeling Search Interaction using Behavioral Economics, ACM, Sheffield, 2024.

[7] N. J. Belkin, R. N. Oddy, and H. M. Brooks, ASK for information retrieval: Part I. Background and theory, Journal of Documentation, London, 1982.

[8] S. Benzarti, W. Tebourski, and W. B. A. Karaa, Toward a Deep Multimodal Interactive Query Expansion for Healthcare Information Retrieval Effectiveness, Springer, Kitakyushu, 2024.

[9] M. Breja and S. K. Jain, Why-Type Question to Query Reformulation for Efficient Document Retrieval, IGI Global, Hershey, 2022.

[10] A. Z. Broder, A taxonomy of web search, ACM, New York, 2002.

[11] C. Buckley, G. Salton, J. Allan, and A. Singhal, Automatic query expansion using SMART: TREC 3, NIST, Gaithersburg, 1995.

[12] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson, Selecting good expansion terms for pseudo-relevance feedback, ACM, Singapore, 2008.

[13] H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li, Towards context-aware search by learning a very large variable length hidden markov model from search logs, ACM, Madrid, 2009.

[14] B. Carterette, V. Pavlu, H. Fang, and E. Kanoulas, Million Query Track 2009 Overview, NIST, Gaithersburg, 2009.

[15] J. Chen, J. Mao, Y. Liu, F. Zhang, M. Zhang, and S. Ma, Towards a Better Understanding of Query Reformulation Behavior in Web Search, ACM/IW3C2, Ljubljana, 2021.

[16] L. Chen, Understanding and exploiting user intent in community question answering, Birkbeck, University of London, UK, 2014.

[17] P.-J. Cheng, C.-H. Tsai, C.-M. Hung, and L.-F. Chien, Query taxonomy generation for web search, ACM, Virginia, 2006.

[18] B. Choi, J. Arguello, and R. Capra, Understanding Procedural Search Tasks "in the Wild", ACM, Austin, 2023.

[19] S.-L. Chuang and L.-F. Chien, Automatic query taxonomy generation for information retrieval applications, Emerald, Bingley, 2003.

[20] M. Dehghani, S. Rothe, E. Alfonseca, and P. Fleury, Learning to attend, copy, and generate for session-based query suggestion, ACM, Singapore, 2017.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Association for Computational Linguistics, Minneapolis, 2019.

[22] D. Downey, S. Dumais, and E. Horvitz, Heads and tails: studies of web search with common and rare queries, ACM, Amsterdam, 2007.

[23] A. Fan, M. Lewis, and Y. N. Dauphin, Hierarchical Neural Story Generation, Association for Computational Linguistics, Melbourne, 2018.

[24] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, The Vocabulary Problem in Human-System Communication, ACM, New York, 1987.

[25] F. X. Han, D. Niu, H. Chen, K. Lai, Y. He, and Y. Xu, A deep generative approach to search extrapolation and recommendation, ACM, Anchorage, 2019.

[26] S. Hirsch, I. Guy, A. Nus, A. Dagan, and O. Kurland, Query Reformulation in E-Commerce Search, ACM, Virtual/China, 2020.

[27] S. M. Hosseini, N. Arabzadeh, M. Zihayat, and E. Bagheri, Enhanced Retrieval Effectiveness through Selective Query Generation, ACM, Birmingham, 2024.

[28] S. N. A. Ibrahim, A. Selamat, and M. H. Selamat, Query optimization in relevance feedback using hybrid GA-PSO for effective web information retrieval, IEEE, Kuala Lumpur, 2009.

[29] I.-H. Kang and G.-C. Kim, Query type classification for web document retrieval, ACM, Toronto, 2003.

[30] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, CTRL: A Conditional Transformer Language Model for Controllable Generation, arXiv, online, 2019.

[31] K. Keyvan and J. X. Huang, How to Approach Ambiguous Queries in Conversational Search: A Survey of Techniques, Approaches, Tools, and Challenges, ACM, New York, 2023.

[32] O. Khattab and M. Zaharia, ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT, ACM, Virtual/China, 2020.

[33] O.-W. Kwon, M.-C. Kim, and K.-S. Choi, Query expansion using domain-adapted, weighted thesaurus in an extended Boolean model, ACM, Maryland, 1994.

[34] H. Li and J. Xu, Semantic Matching in Search, Now Publishers, Delft, 2014.

[35] S. Manchanda, M. Sharma, and G. Karypis, Intent term selection and refinement in e-commerce queries, arXiv, online, 2019.

[36] X. Mao, S. Huang, R. Li, and L. Shen, Automatic keywords extraction based on co-occurrence and semantic relationships between words, IEEE, New York, 2020.

[37] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, The Natural Language Decathlon: Multitask Learning as Question Answering, arXiv, online, 2018.

[38] R. Mitra, M. Gupta, and S. Dandapat, Transformer Models for Recommending Related Questions in Web Search, ACM, Virtual/Ireland, 2020.

[39] H. Moen, L.-M. Peltonen, H. Suhonen, H.-M. Matinolli, R. Mieronkoski, K. Telen, K. Terho, T. Salakoski, and S. Salanterä, An Unsupervised Query Rewriting Approach Using N-gram Co-occurrence Statistics to Find Similar Phrases in Large Text Corpora, Linköping University Electronic Press, Turku, 2019.

[40] Y. L. Narayanan and H. Fani, RePair: An Extensible Toolkit to Generate Large-Scale Datasets for Query Refinement via Transformers, ACM, Birmingham, 2023.

[41] W. Chen, F. Cai, H. Chen, and M. de Rijke, Personalized Query Suggestion Diversification, In Proceedings of the ACM SIGIR Conference, ACM, 2017.

[42] N. Fiorini and Z. Lu, Personalized Neural Language Models for Real-World Query Auto Completion, In Proceedings of the NAACL-HLT Conference, 2018.

[43] R. Nogueira, J. Lin, and A. I. Epistemic, From doc2query to docTTTTTquery, Online, 2019.

[44] R. F. Nogueira, W. Yang, J. Lin, and K. Cho, Document Expansion by Query Prediction, arXiv, online, 2019.

[45] J. M. Ponte and W. B. Croft, A Language Modeling Approach to Information Retrieval, ACM, Melbourne, 1998.

[46] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever et al., Language models are unsupervised multitask learners, OpenAI, San Francisco, 2019.

[47] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, JMLR, Cambridge, 2020.

[48] D. Rajaei, Z. Taheri, and H. Fani, No Query Left Behind: Query Refinement via Backtranslation, ACM, Boise, 2024.

[49] S. E. Robertson and H. Zaragoza, The Probabilistic Relevance Framework: BM25 and Beyond, Now Publishers, Delft, 2009.

[50] J. J. Rocchio, Relevance feedback in information retrieval, Prentice-Hall, Englewood Cliffs, 1971.

[51] C. D. Schultz, Informational, transactional, and navigational need of information: relevance of search intention in search engine advertising, Springer, New York, 2020.

[52] C. Shah, R. W. White, R. Andersen, G. Buscher, S. Counts, S. S. S. Das, A. Montazer, S. Manivannan, J. Neville, X. Ni, N. Rangan, T. Safavi, S. Suri, M. Wan, L. Wang, and L. Yang, Using Large Language Models to Generate, Validate, and Apply User Intent Taxonomies, arXiv, online, 2023.

[53] S. Shekarpour, E. Marx, S. Auer, and A. P. Sheth, RQUERY: Rewriting Natural Language Queries on Knowledge Graphs to Alleviate the Vocabulary Mismatch Problem, AAAI Press, San Francisco, 2017.

[54] A. A. Shiri, End-user interaction with thesaurus-enhanced search interfaces, an evaluation of search term selection for query expansion, University of Alberta, Edmonton, 2003.

[55] A. Singhal et al., Modern information retrieval: A brief overview, IEEE, New York, 2001.

[56] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie, A hierarchical recurrent encoder-decoder for generative context-aware query suggestion, ACM, Melbourne, 2015.

[57] M. Tamannaee, H. Fani, F. Zarrinkalam, J. Samouh, S. Paydar, and E. Bagheri, ReQue: A Configurable Workflow and Dataset Collection for Query Refinement, ACM, Virtual/Ireland, 2020.

[58] T. Tao and C. Zhai, Regularized estimation of mixture models for robust pseudo-relevance feedback, ACM, Seattle, 2006.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is All you Need, NeurIPS, 2017.

[60] J. Xu and W. B. Croft, Quary expansion using local and global document analysis, ACM SIGIR Forum, New York, 2017.

[61] D. Yang, Y. Zhang, and H. Fang, An Exploration Study of Mixed-initiative Query Reformulation in Conversational Passage Retrieval, NIST, Online, 2022.

[62] G. Zerveas, R. Zhang, L. Kim, and C. Eickhoff, Brown University at TREC Deep Learning 2019, arXiv, 2020.

[63] X. Zhang, Improving personalised query reformulation with embeddings, Journal of Information Science, 2022.

[64] L. Zhao and J. Callan, Term necessity prediction, In Proceedings of the 19th ACM CIKM, 2010.

[65] Y. Zhou, Z. Dou, and J.-R. Wen, Encoding history with context-aware representation learning for personalized search, In Proceedings of the 43rd ACM SIGIR, 2020.

[66] Y. Zhu, L. Pang, Y. Lan, H. Shen, and X. Cheng, LoL: A comparative regularization loss over query reformulation losses for pseudo-relevance feedback, In Proceedings of the 45th ACM SIGIR, 2022.

[67] E. Rencis, Application of a configurable keywords-based query language to the healthcare domain, Journal of Advances in Information Technology, 2021.