It's All Relative! – A Synthetic Query Generation Approach for Improving Zero-Shot Relevance Prediction

Aditi Chaudhary **Karthik Raman** **Michael Bendersky**
Google Research, USA
{aditichaud, karthikraman, bemike}@google.com

Abstract

Large language models (LLMs) have shown promising ability to generate synthetic query-document pairs by prompting with as few as 8 demonstrations (Dai et al., 2022). This has enabled building better IR models, especially for tasks with no training data. Typically, such synthetic query generation (QGen) approaches condition on an input context (e.g. a text document) and generate a query relevant to that context, or condition the QGen additionally on the relevance label (e.g. relevant vs irrelevant) to generate queries across relevance buckets. However, we find that such QGen approaches are sub-optimal as they require the model to reason about the desired label and the input from a handful of examples. In this work, we propose to reduce this burden of LLMs by *generating queries simultaneously for different labels*. We hypothesize that instead of asking the model to generate, say, an irrelevant query given an input context, asking the model to generate an irrelevant query *relative to a relevant query* is a much simpler task. Extensive experimentation across nine IR datasets shows that synthetic queries generated in such a fashion translates to better downstream performance.

1 Introduction

Recently, Large Language Models (LLMs) (GPT-3:Brown et al. (2020), PaLM: Anil et al. (2023), LLAMA-2 Touvron et al. (2023)) have been applied to automatically generate task-specific data which can be used to train downstream models (section 6). For example, Promptagator (Dai et al., 2022) and InPars (Bonifacio et al., 2022) use as few as 8 task-specific demonstrations and prompt the LLM to generate synthetic queries for new documents. These demonstrations are pairs of input-output examples where the input is a document text and the output is a query relevant to that context. However, this process only generates "relevant" synthetic queries. To create the corresponding negative or "irrelevant" pairs, a retriever is used to retrieve documents for each synthetic query, from which the hard negatives are constructed. Finally, a task-specific downstream model is trained using this synthetic data. Chaudhary et al. (2023) forgo the retriever step, and instead generate queries across different relevance buckets (e.g. relevant and irrelevant) by providing the relevance label along with the document as the context. Although, they find conditioning query generation (QGen) on the different relevance labels does outperform approaches such as Promptagator, the downstream model trained on the task-specific synthetic data is outperformed by the traditional transfer learning model, where a model is trained on a related dataset and directly applied to the target task.

In this work, we propose a novel few-shot QGen approach where we prompt the model to *generate queries across different relevance labels relative to each other*. For example, given an input document and two relevance labels (relevant and irrelevant), the model is required to generate both a relevant query and an irrelevant query. By forcing the model to generate an irrelevant query after generating a relevant query, forces the model to condition on the previously generated query along with the document context (Figure 1). We hypothesize this is a much more simpler task for the model in comparison to generating, say, an irrelevant query from only the document context or the label alone. This is inspired from the observation that both humans and LLMs find the task of comparing two things with respect to each other a simpler task, as opposed to judging things in isolation (Christiano et al., 2017; Qin et al., 2023) and such a binary comparison protocol is specially actively applied for LLM alignment (Touvron et al., 2023). To evaluate our hypothesis, we conduct zero-shot experiments with IR datasets from the BEIR (Thakur et al., 2021) benchmark. We generate synthetic

Findings of the Association for Computational Linguistics: NAACL 2024, pages 1645–1664

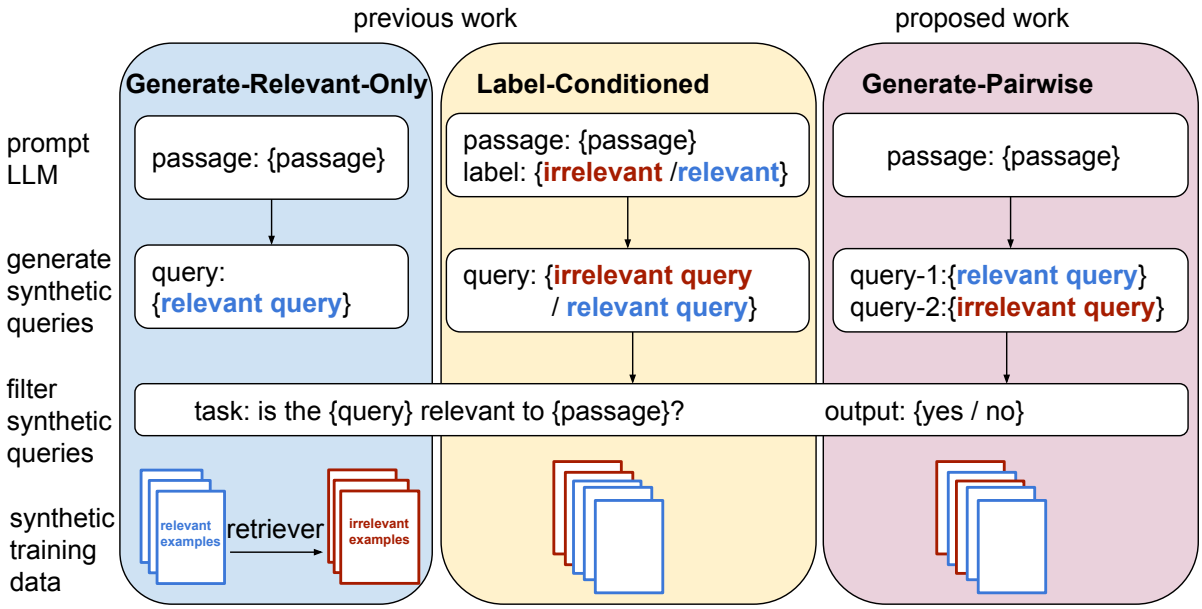

Figure 1: An overview of our proposed pairwise query generation approach (GENERATE-PAIRWISE), where we prompt the model to generate both the relevant and the irrelevant query at the same time, forcing the model to condition the irrelevant query generation on the previous generated query. We compare the proposed approach with previous QGen approaches, GENERATE-RELEVANT-ONLY, where only relevant queries are generated followed by a retriever to retrieve irrelevant examples and, LABEL-CONDITIONED, where the query generation is conditioned on one relevance label, at a time, to generate a query. Next, the generated queries are filtered automatically to ensure self-consistency. and the resulting synthetic data is used for downstream training.

data for each task by prompting the LLM with only 10 examples from MS-MARCO (Nguyen et al., 2016) and train a downstream model. We use the downstream task performance as a signal to evaluate the effectiveness of the QGen model and our key observations are:

- Generating queries relative to each other leads to an improved downstream score – (+8.6 NDCG@10) avg. improvement over existing QGen approaches for five out of six BEIR datasets. (Table 1)

- For three datasets, the downstream model trained only on our generated queries even outperforms the strong skyline of transfer learning model which is fine-tuned on $\sim 6M$ MS-MARCO examples and directly applied to the target task. For two datasets, the best QGen model is only 3 points behind this strong skyline model, suggesting the generated data is almost at par with human-labeled data.

- A case study on fine-grained relevance prediction shows that generating pairwise queries is also well-suited to generate nuanced queries across multiple relevance labels (Table 5).

2 Synthetic Query Generation

We follow Promptagator's few-shot query generation (QGen) setup, which comprises of three steps, namely, prompt-based query generation, query filtration, and, downstream model training.

2.1 Background: Prompt-Based QGen

The first step of the QGen process generates task-specific query-document pairs using an instruction prompt p and the target task T's document corpus $\mathcal{D}_\mathcal{T}$. Existing approaches such as Promptagator and InPars generate relevant query-document pairs. Specifically, Promptagator use k task-specific relevant query-document pairs as the instruction prompt $p_T = \{(d_i, q_i)^k, d_t\}$, where the document d_i is relevant to the query q_i, and d_t denotes the new document for which we want to generate the query. The prompt p_T is then run on the target task corpus $\mathcal{D}_T$ to generate a large relevant query-document set. Chaudhary et al. (2023) condition the QGen model on the relevance label as well, which allows them to generate query-document pairs across different relevance labels ($l \in L$), where L is the set of all labels. The instruction prompt $p_T = \{(l_i, d_i, q_i)^k, (l_t, d_t)\}$ comprises of k query-document-label triplets, where for the doc-

ument d_i the query q_i's relevance label is l_i. To generate a new query, the prompt takes the desired label (l_t) along with the document (d_t), generating a query whose relevance level with the document is implied by the label.

2.2 Proposed: Pairwise Query Generation

Both the label-conditioned (Chaudhary et al., 2023) and the non-label conditioned (Promptagator), generate queries in isolation of other queries. As motivated in the introduction, comparing two outputs is an easier setup as opposed to evaluating an output in isolation (Christiano et al., 2017; Qin et al., 2023). This makes us wonder whether formulating the QGen task in a similar fashion would lead to high-quality synthetic data.

Specifically, we propose pairwise query generation, where given an input document, the model is required to generate both the relevant and the irrelevant query. Like before, we construct an instruction prompt p using k input-output examples, where the input is a document d and and the output is a pair of queries $(q_{\text{rel}}, q_{\text{irrel}})$ where the first query q_{rel} is relevant to the d while the second query q_{irrel} is irrelevant. The pairwise prompt, therefore, is as follows: $p = \{(d_i, q_{\text{rel}}, q_{\text{irrel}})^k, d_t\}$. Given a new document, the model is expected to generate both the relevant and the irrelevant queries at the same time, as shown in Table 7. By forcing the model to generate both queries, the generation of the irrelevant query is conditioned on the previously generated relevant query along with the input context. Note, the generated irrelevant query will be thematically similar to the input document and not completely unrelated. In other words, these queries are "hard negatives", which previous works (Gao et al., 2021; Karpukhin et al., 2020; Guu et al., 2020) have shown to be better for training downstream IR models rather than using randomly sampled or in-batch negatives.

2.3 Query Filtration

After generating the synthetic queries, round-trip consistency filter (Alberti et al., 2019) is applied to remove noisy queries. For example, Promptagator use a retriever trained on the synthetic data to check whether the generated query is relevant to the document from which it was generated, and find that including the round-trip filtering is important for improving the query quality. Given the superiority of LLMs, we prompt the LLM again, but this time to predict whether the generated synthetic query q_{syn} is relevant or not to the document d_t from which it was generated. Specifically, the prompt comprises of k query-document example pairs across both relevance labels: $p_{\text{filter}} = \{(q_i, d_i, l_i)^k, (q_{\text{syn}}, d_t)\}$, where $l \in L = \{\text{relevant}, \text{irrelevant}\}$. We apply the p_{filter} on the synthetic query-document pair (q_{syn}, d_t) and compute the log-likelihood to score each output label L, given the prompt. Then, we select the label with the highest score, as shown below: $l_{\text{pred}} = \arg\max_L \text{score}(l, p_{\text{filter}})$ where p refers to the filtration prompt and *score* is the log-likelihood. We only retain those queries whose predicted label l_{pred} matches the label for which the query was generated. For instance, in our proposed pairwise QGen, the first query will be retained only if it is rated as relevant and, similarly, the second query if it is rated as not relevant. In addition to the round-trip filtering, we also remove duplicate queries, i.e. queries that got labeled with different relevance labels for the same document, following Chaudhary et al. (2023).

2.4 Downstream Model Training

Finally, a downstream model is trained on the filtered synthetic data. Promptagator train a task-specific retriever and a re-ranker on the synthetic query-document pairs as their end-task is to improve document retrieval. Similar to Chaudhary et al. (2023) we train a relevance prediction model, since our goal is to improve zero-shot relevance prediction. Note that through our proposed method, the model is able to generate both relevant and irrelevant examples, alleviating the need of running the additional step of an retriever, as done by Dai et al. (2022) and Bonifacio et al. (2022).

3 Experimental Setup

Our goal is to improve relevance prediction for tasks with no training data, for which we generate sufficient synthetic data and train a downstream model. We experiment with six English IR tasks from the BEIR benchmark (Thakur et al., 2021), under zero-shot settings, where we generate query-document examples for each task. We construct the prompt using a fixed set of examples from the MS-MARCO dataset (Nguyen et al., 2016), which covers a variety of domains, and generate new queries using the respective target task's document corpora.[1] We summarize all the datasets in

[1]During the review process, a reviewer raised questions about the use of term 'zero-shot' for our experimental setting. In many of the public datasets (e.g. BEIR, MIRACL, MS-MARCO, etc), usually a large document corpora is pro-

subsection A.1.

3.1 Model

QGen Model For both the query generation and query filtration, we use PaLM-2 (M) (Anil et al., 2023), a multilingual LLM pretrained on a variety of corpora such as web documents, books, code, mathematics and conversational data, to name a few. For some datasets the provided document corpora is quite large, thus, we randomly sample 50,000 documents for each task to apply the QGen model, except for TREC-COVID where we use all the documents to measure the effect of using the entire corpus.

Downstream Model To evaluate the quality of the generated queries, we train a pointwise encoder using mt5-XXL (Xue et al., 2021) for relevance prediction. We use all of the filtered synthetic data as our training split. As validation split, we sample 5k examples (2,500 for each binary relevance label) from MS-MARCO, and use the test split of the target task to report the final metric. Similar to Chaudhary et al. (2023), we use the NDCG metric to report the downstream model performance.[2] NDCG measures the ranking performance i.e. whether the model is able to rank true relevant documents higher than any other document. For some datasets such as TREC-COVID, Touché, and DBPedia, the test split already contains examples of not relevant and partially relevant query-document examples, in addition to the relevant examples. However, for the other three datasets, the test split only provides relevant examples, assuming every other document to be irrelevant. Therefore, for such datasets, following Zhuang et al. (2022) and Dai et al. (2022), we use BM25 to retrieve top-20 documents for each test query and combine that with gold-annotated relevant examples. Please refer to subsection A.2 for the hyperparameter settings for the QGen model and the downstream model.

vided separately in addition to the train/valid/test qrels (i.e. the query-document relevance judgements). The documents covered in the train/valid/test are a subset of this large corpora. And often it happens that even across this gold-standard train/valid/test files, there is a significant document overlap (of course the queries are unique across the train/valid/test). In our setting, for QGen, we randomly choose documents from this separately provided corpora. As in a zero-shot setting, we assume we have no information about the test set i.e. we do not know what queries or documents are annotated in the test set. Hence, from that perspective, we refer to the evaluation setting as zero-shot.

[2] As explained in Section 4.3 of Chaudhary et al. (2023), using NDCG over accuracy avoids the need for mapping labels across different datasets, as many test datasets do not have the same label set as the train dataset.

3.2 Baselines

Transfer Learning Fine-tuning a model on a related dataset and applying it as-is on a new domain, is an effective strategy to improve zero-shot performance. Specifically, for all BEIR datasets, we fine-tune a mt5-XXL model on the general purpose MS-MARCO dataset. We sample 6 million query-document examples, equally distributed across both relevant and irrelevant labels. We treat this model as a *skyline* as tit has access to all training data while for QGen models only 10 examples are used.

Generate-Relevant-Only We prompt the LLM to generate relevant-only queries, similar to Promptagator (Dai et al., 2022), where our prompt consists of 10 MS-MARCO examples of relevant query-document pairs and the LLM generates relevant queries for a new document. Next, a filtration prompt is applied to remove queries that are not relevant to the document from which they were generated. We use a T5-based retriever (Ni et al., 2021, 2022) to retrieve hard negatives i.e. one irrelevant document for each synthetic query, similar to Promptagator. Since our main goal is to evaluate the quality of different QGen models, we use a general-purpose retriever while Promptagator train a task-specific retriever.

Label-Conditioned Following Chaudhary et al. (2023), we prompt the model to generate a query by using the required relevance label and the document as context. Specifically, the prompt includes 5 relevant and 5 irrelevant MS-MARCO query-document examples. For a new task document, we then generate synthetic queries for both the relevant and irrelevant label. Next, we prompt the LLM with the generated query and document to predict the relevance label and filter those queries whose desired label does not match the predicted label. Since the model already generates irrelevant examples, we directly train the downstream model on the above data, without running the retriever.

3.3 Proposed Models

We experiment with the following two model variants under the pairwise query generation approach.

Generate-Pairwise We prompt the model to generate both the relevant and the irrelevant query for

		Gold Negatives			Gold + BM25 Negatives		
	Model	**TREC-COVID (all)**	**Touché (50k)**	**DBPedia (50k)**	**Climate-Fever (50k)**	**FIQA (50k)**	**FEVER (50k)**
Skyline	Transfer Learning	0.7615	0.7714	<u>0.6181</u>	0.5129	<u>0.6781</u>	<u>0.9041</u>
Baseline	Generate-Relevant-Only	0.4145	0.6512	0.5471	0.4704	0.6027	**0.7915**
	Label-Conditioned	0.7765	0.7466	0.5152	0.4888	0.4005	0.7032
Ours	Generate-Pairwise	0.7831	**0.7881**	**0.5631**	**0.5562**	0.4769	0.7235
	Iterative-Pairwise	**0.8004**	0.7762	0.5557	0.4972	**0.6584**	0.7054

Table 1: We compare the NDCG@10 metric on the test split of all tasks (higher the better). 'Gold Negatives' refer to those datasets where the negative examples are also provided by the BEIR benchmark, while 'Gold+BM25 Negatives' refer to the datasets where no gold negatives are provided. For such datasets, BM25 is used to retrieve top-20 documents which are combined with the gold positive examples (i.e. relevant documents), over which the NDCG@10 is reported. The number in the brackets below each dataset refers to number of documents used for query generation. Overall best scores are <u>underlined</u> and the best scores among the QGen models are **highlighted**.

a given document, at the same time. Specifically, we prompt the model with the same 10 relevant query-document examples as above, but for each relevant query-document pair, we additionally also add an irrelevant query, as shown in Table 7. We generate two outputs for each document, which in this case means two relevant and two irrelevant queries. Next, we filter the queries using the filtration prompt and train the downstream model. Similar to the LABEL-CONDITIONED model, the retriever step is not applied as both relevant and irrelevant examples are generated.

Iterative-Pairwise Similar to the GENERATE-RELEVANT-ONLY model, we first generate relevant-only queries for each document and filter queries that are rated as not relevant. Next, we prompt the LLM again with the same document context along with the filtered relevant query and prompt the model to generate an irrelevant query, using the same prompt as GENERATE-PAIRWISE model. Then, we apply the filtration prompt on the generated irrelevant queries, and remove queries that are rated as relevant. This is similar in principle to the above variant, where we still condition the generation of the irrelevant query on the previous generated relevant query. Like the above model, for every QGen step we generate two outputs.

In Table 10–Table 12 we show the amount of synthetic query-document examples generated for different tasks across QGen models. Specifically, we show the number of queries generated and filtered after each QGen step, for each of the four QGen models.

4 Results and Discussion

We report our main results in Table 1 and find that among all QGen approaches, our proposed pairwise method results in the best downstream performance for five out of the six tasks. Specifically, the average improvement of GENERATE-PAIRWISE over existing baselines is +5.6 NDCG@10 points and that of ITERATIVE-PAIRWISE is +7.3 points. We observe that even conditioning the QGen model on the relevance label (LABEL-CONDITIONED) in addition to the document context already outperforms the GENERATE-RELEVANT-ONLY model by +2 NDCG@10 points (avg. across all tasks), which is in-line with Chaudhary et al. (2023). The proposed pairwise models further improves upon LABEL-CONDITIONED by +7 NDCG@10 points (avg.) across all tasks, suggesting that conditioning on a previous generated query provides an even stronger signal to the LLM to directly generate ranked query-document pairs, which aligns with our downstream task. Interestingly, we find that for three of the datasets (TREC-COVID, Touché, and Climate-FEVER), the models solely trained on synthetic data even outperforms the transfer learning skyline, where the entire MS-MARCO data was used for fine-tuning. This is unlike Chaudhary et al. (2023), where none of the QGen approaches outperformed the simple but strong transfer learning skyline. What makes this result interesting is that in all of the QGen models, only 10 MS-MARCO examples are used, highlighting again that a) *quality matters over quantity* i.e. LLMs can learn to generate high-quality data given a good set of exemplars, and b) even noisy in-domain training data is effective than out-of-domain gold annotated data.

QGen Prompt Source	QGen Model	NDCG@10
MS-MARCO	Label-Conditioned	40.05
	Generate-Pairwise	47.69
FIQA	Label-Conditioned	45.44
	Generate-Pairwise	**56.45**

Table 2: We compare the effect of using task-specific exemplars (FIQA) in the QGen prompt over the generic MS-MARCO exemplars.

However, for the FEVER task, GENERATE-PAIRWISE is not the best performing model. This is probably because there is a mis-match between the irrelevant queries of the test split and the generated irrelevant queries. Since FEVER is a fact-verification task, the irrelevant queries of the test split (derived from sentence claims) are entity-centric while the generated queries are more general or concept-focused (check Table 14 for examples of generated queries). This is because the model is copying the style of MSMARCO exemplars used in the QGen prompt. GENERATE-RELEVANT-ONLY is not affected because the irrelevant examples are derived from a retriever, i.e. for every generated relevant query the top-retrieved document forms the irrelevant example. Since most generated relevant queries are already entity-centric, the irrelevant examples align well with the test split distribution. We conduct ablation experiments to better understand the pairwise generation.

Do the trends hold if task-specific exemplars are used in the QGen prompt? Yes! As mentioned above, we choose to use the same set of exemplars from MS-MARCO to generate queries for all tasks, while Promptagator use 8 task-specific exemplars. Therefore, for one of the tasks, FIQA-2018, we re-run two QGen models, namely LABEL-CONDITIONED and GENERATE-PAIRWISE, but this time constructing all prompts using 10 exemplars from the training corpus of FIQA-2018. From Table 2, we find for both the QGen models using task-specific exemplars leads to a gain in the performance (+5 points for LABEL-CONDITIONED model and +8 points for GENERATE-PAIRWISE). This is expected given that the LLM is now given in-domain knowledge to start from, however, with this ablation we wanted to highlight that the gains of the pairwise approach still hold.

Do the trends hold for smaller downstream models? Yes! For many real-world applications inference matters more than fine-tuning, since inference needs to run several times and often on large set of documents.[3] Therefore, we aim to examine the QGen performance across smaller downstream models. In Table 3, we compare the downstream performance of three mt5-* models, namely, mt5-Large (1.2B), mt5-XL (3.7B), and mt5-XXL (13B) on four tasks.[4] We find that the trends across all model sizes are consistent, with the proposed pairwise QGen outperforming other models. This is an important result as when there is insufficient compute or the requirement is to rank millions of documents, smaller models similarly benefit from the proposed QGen models' generated queries.

Do the trends hold for multilingual settings? To some extent! We conduct a limited study with the multilingual MIRACL (Zhang et al., 2023) benchmark, using the English and Hindi datasets (details in subsection A.1). From Table 4 we see that GENERATE-PAIRWISE outperforms all the baselines for the English dataset, while for Hindi it is only 1.2 NDCG@10 points behind the best performing LABEL-CONDITIONED. Interestingly, we find that *all* QGen models outperform the skyline by a significant margin, including the skyline trained on the in-domain MIRACL training portions. This is because the in-domain gold training data is limited in size (30k judgements for English and 11k for Hindi), while the generated training data is twice the quantity (Table 8).

Which proposed variant to prefer? Depends on computation!. From the above results, it is clear that our proposed variants, either the GENERATE-PAIRWISE or the ITERATIVE-PAIRWISE, outperform the existing QGen baselines on the downstream task. Given that both our proposed variants have the same governing principle behind the query generation process, a natural question then arises *when should one variant be preferred over the other*.[5] We recommend choosing the approach based on the available computation. For instance, the GENERATE-PAIRWISE generates the relevant and irrelevant queries in a single iteration, followed by a combined query filtration which is computation friendly, whereas for ITERATIVE-PAIRWISE variant, we have to undergo two steps of query generation and two steps of fil-

[3] https://shorturl.at/jkAUZ

[4] https://github.com/google-research/multilingual-t5

[5] We thank the reviewer for asking this question, which prompted us to add the following paragraph in the paper.

Model Size	Type	QGen Model	Touché	DBPedia	Climate-Fever	FIQA
XXL	Baseline	Generate-Relevant-Only	0.6512	0.5471	0.4704	0.6027
		Label-Conditioned	0.7466	0.5152	0.4888	0.4005
	Ours	Generate-Pairwise	**0.7881**	**0.5631**	**0.5562**	0.4769
		Iterative-Pairwise	0.7762	0.5557	0.4972	**0.6584**
XL	Baseline	Generate-Relevant-Only	0.6646	0.5153	0.4011	0.4636
		Label-Conditioned	0.7466	0.5152	0.4888	0.4005
	Ours	Generate-Pairwise	**0.7958**	0.4942	**0.5026**	0.3604
		Iterative-Pairwise	0.7903	**0.5284**	0.4667	**0.5099**
L	Baseline	Generate-Relevant-Only	0.6793	0.5102	0.3959	0.4093
		Label-Conditioned	0.7835	0.3573	0.4033	0.2288
	Ours	Generate-Pairwise	**0.8249**	0.4871	**0.4278**	0.2684
		Iterative-Pairwise	0.8070	**0.4927**	0.4200	**0.4568**

Table 3: Comparing the performance of the downstream models trained on the generated queries across different sizes. Note here the generated queries for each downstream model is still from the PaLM-2 (M) model, we vary the downstream model size and measure the NDCG@10 (higher the better).

	QGen Model	en	hi
Skyline	Transfer Learning (MS-MARCO)	0.5932	-
	Transfer Learning (MIRACL)	0.5891	0.5449
Ours	Generate-Relevant-Only	0.7126	0.7123
	Label-Conditioned	0.6225	**0.8036**
	Generate-Pairwise	**0.7964**	0.7912
	Iterative-Pairwise	0.7899	0.7912

Table 4: We compare the NDCG@10 performance on the MIRACL English and Hindi datasets.

	QGen Model	NDCG@10
Skyline	Transfer Learning*	0.8927
Baseline	FINETUNE-BASED-LABELCOND (Chaudhary et al., 2023)	0.8553
	Generate-Relevant-Only	0.6112
	Label-Conditioned	0.8779
Ours	Generate-Pairwise	**0.8882**
	Generate-AllLabels	0.8700

Table 5: We compare the effect of using QGen for the fine-grained relevance prediction task. We include the best performing QGen model from Chaudhary et al. (2023) (FINETUNE-BASED-LABELCOND), a mt5-XXL model fine-tuned for the query generation task using relevance labels and documents as context. *The skyline results are used from Chaudhary et al. (2023).

tration, once for the relevant query generation and next for the irrelevant query generation. Since the latter is more selective, it results in fewer number of total queries compared to the former (see Table 9–Table 12) which could be also one reason why ITERATIVE-PAIRWISE is sometimes slightly behind GENERATE-PAIRWISE. This probably also explains why Iterative-Pairwise leads to better downstream performance for FIQA and TREC-COVID datasets. Compared to other datasets, FIQA and TREC-COVID are the furthest from the source domain (i.e. MS-MARCO) used for constructing the prompt exemplars, which suggests that the queries generated might not be of great quality and therefore the ITERATIVE-PAIRWISE method being more selective, leads to higher performance.

For the above experiments, we generated queries for binary labels. However, some tasks have nuanced relevance judgements, for instance, TREC datasets are often annotated on 4-point scale. *Could we then generate queries for such nuanced labels using the above QGen approaches?*.

5 Case Study: QGen for Fine-grained Relevance Prediction

We follow Chaudhary et al. (2023) and conduct a limited study where we use ESCI (Reddy et al., 2022), a shopping relevance dataset, to generate training data for another shopping dataset WANDS (Chen et al., 2022). Unlike the BEIR benchmark, where there were only two relevance labels, both ESCI and WANDS are multi-class and have nuanced relevance labels (details in subsection A.1).

In this setting, we construct prompts for all the QGen models using examples from ESCI. For example, for the GENERATE-RELEVANT-ONLY, we use the same setting as described in subsection 3.2,

with the difference that instead of selecting exemplars from MS-MARCO, we select 10 query-product pairs from ESCI with the relevance label as *exact*, and as before, we use a retriever to retrieve the hard negatives. For all the other models, we generate queries across all relevance labels. For example, for the LABEL-CONDITIONED model, where we condition the query generation on the label, we select 10 exemplars, ensuring at least two examples for each of the four relevance labels are included. Next, we generate queries for each WANDS product and each of the four relevance labels. For our proposed GENERATE-PAIRWISE model, earlier, we prompted the model to generate a relevant query followed by an irrelevant query. Given that in this case there are four relevance labels, we adapt the task of pairwise generation to also take into account for which two labels the queries need to be generated. Specifically, we prompt the model with the product and an instruction outlining for which two of the four labels the model should generate the queries (Table 6).[6] Next, to generate new queries for each WANDS product, we create four label-combinations, namely, exact-complement, complement-exact, substitute-irrelevant, and irrelevant-substitute.[7] We also extend the above model to generate queries for all labels at the same time GENERATE-ALLLABELS. In this setting, we prompt the model to also generate queries in the decreasing order of relevance. As before, for each setting filter the queries whose predicted label does not match the label for which it was generated (details in subsection A.3). The exact prompt formats are outlined in Table 6.

5.1 Results

In Table 5 we find that among all QGen models, GENERATE-PAIRWISE performs the best. Chaudhary et al. (2023) found that a model (mt5-XXL) fine-tuned for query-generation using labels as conditioning context outperformed the prompt-based LABEL-CONDITIONED QGen model. Interestingly, we find here that our prompt-based LABEL-CONDITIONED model already outperforms Chaudhary et al. (2023)'s best model, which was fine-tuned on millions of ESCI training examples, highlighting that PaLM-2 (M) model is a strong foundational model even when used in a few-shot prompt fashion. Within our proposed pairwise generation, we find that requiring the model to generate queries for all four labels (GENERATE-ALLLABELS) leads to only 62k training examples as opposed to 168k examples generated by restricting the model to generate queries for only two labels. This is because nearly 46% of the LLM outputs were incorrectly formatted due to which the queries could not be parsed.[8] This is probably why the performance of GENERATE-ALLLABELS is lower than GENERATE-PAIRWISE. From Table 15, we see that the generated queries for GENERATE-PAIRWISE are indeed in the decreasing order of relevance, while both the relevant and not-relevant queries for GENERATE-RELEVANT-ONLY are in fact relevant to the product, which probably explains the poor performance of the latter model.

6 Related Work

LLMs have been extensively explored for synthetic text generation across different tasks, such as, Structured Prediction (Chen et al., 2023), Text Classification (Gao et al., 2022; Meng et al., 2022; Gupta et al., 2023; Wang et al., 2023; Yu et al., 2023b), Information Retrieval (Dai et al., 2022; Bonifacio et al., 2022), to name a few.

LLMs for Query Generation As mentioned above, Promptagator (Dai et al., 2022) and InPars (Bonifacio et al., 2022) both generate relevant queries, by prompting the model with relevant examples. InPars also propose a second prompt variant, called as GBQ (Guided By Bad Questions), where the LLM is shown a bad question or an irrelevant question for every relevant query-document example, similar to how we show irrelevant query for every relevant example. However, InPars discard the generated irrelevant questions and use a retriever to retrieve irrelevant examples. Another point of difference between Promptagator, InPars and our work is the type of downstream training. Because our main focus is to compare different QGen approaches, we use a simple pointwise objective function i.e. pose the task as binary classifi-

[6] We prioritize including those label combinations where the first label's relevance is higher than the second label's relevance, for example, pairs such as exact-substitute, exact-irrelevant. We do include two exemplars where the the first query to be generated is for irrelevant label, ensuring that the model has seen at least one example for all four labels.

[7] There were two reasons for these particular combinations: 1) these combinations ensure that for each product we generate both query1 and query2 for each relevance label, and 2) by skipping adjacent labels we hope to improve the generated output quality as our initial experiments showed that the adjacent labels often do not have clear separation boundary.

[8] Details in subsection A.4

cation, while Promptagator and InPars use ranking-specific objective functions.

Task-Specific Prompting for Synthetic Text Generation The most common prompting strategy for text generation has been to use seed exemplars where the prompt is guided by a label-description (for example, Self-Instruct (Wang et al., 2023) and AttrPrompt (Yu et al., 2023a) for text classification tasks). Specifically, AttrPrompt find that simply conditioning on class often leads to lack of diversity in generated examples and propose to condition on other attributes such as length and style. More recently, Gupta et al. (2023) propose TarGEN to generate synthetic data for text classification tasks, without using seed exemplars. Instead, TarGEN aim to generate diverse examples by incorporating task-specific elements (e.g. linguistic information about the task) in the text generation process. They first generate these task-specific elements (these are not actual exemplars but intermediate hints useful for the task) and similar to Chaudhary et al. (2023) they also condition their example generation on the target label. The generated examples are run through a LLM with a task-specific prompt to remove examples, similar to subsection 2.3. The above works suggest that adding task-specific attributes is a useful prompting strategy in addition to the final task labels, which aligns well with our finding as well, where using a ranking-based conditioning (e.g. GENERATE-PAIRWISE and ITERATIVE-PAIRWISE) is helpful for ranking tasks.

LLMs for Relevance Judgements Complementary to our work, LLMs have also been directly used for relevance judgements or ranking documents (Thomas et al., 2023; Qin et al., 2023; Zhuang et al., 2023; Faggioli et al., 2023) via prompting. Qin et al. (2023) propose the pairwise ranking prompting (PRP) that uses a query and a pair of documents as prompt to the LLMs for ranking. Specifically, they find that instead of asking the model to rank all documents in a list, providing documents in pair simplifies the ranking problem. In addition to using binary relevance labels (relevant and irrelevant), some existing works (Zhuang et al., 2023; Faggioli et al., 2023) also prompt LLMs with intermediate relevance labels (e.g. partially relevant) to improvie fine-grained relevance prediction.

7 Next Steps

In this work, we have shown that by aligning the query generation task with the downstream ranking task, we are able to generate synthetic data useful to the downstream task. In all the tasks, the query generation is conditioned on the document or a product context. However, there are tasks within BEIR such as the Duplicate-Question Retrieval task where the task is retrieving a duplicate or similar question. Given there is no conditioning context, it will be interesting to check how to adapt our best performing pairwise generation for such a task. As we saw in Table 14, the generated irrelevant queries for FEVER were not grounded to the entities in the document. Therefore, we plan to explore approaches to control the LLM generation along multiple aspects such as relevance, diversity, task-specific information. In future, we also plan to train a retriever on the generated synthetic data, similar to Promptagator, further improve the downstream performance.

8 Limitations

In this paper, we evaluate the generated query quality by running the downstream model for each QGen approach, which amounts to running a large set of experiments across different tasks resulting in computation and time cost. We believe that having evaluation techniques which could simulate the downstream performance without requiring to run the downstream models every time, will save a lot of compute and time for researchers exploring the synthetic text generation space.

9 Statement of Ethics

The use of pretrained LLMs for language generation always carries the risk of bias propagation, and since in this case, we are training a downstream model on the generated synthetic data, there is a chance this risk propagates further down the pipeline.

Acknowledgements

We would like to thank Honglei Zhuang for their valuable suggestions and inputs throughout the project. We also thank the reviewers for their careful review and suggestions which helped improve the paper.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Ábrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, and et al. 2023. Palm 2 technical report. *CoRR*, abs/2305.10403.

Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, et al. 2020. Overview of touché 2020: argument retrieval. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*, pages 384–395. Springer.

Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aditi Chaudhary, Karthik Raman, Krishna Srinivasan, Kazuma Hashimoto, Mike Bendersky, and Marc Najork. 2023. Exploring the viability of synthetic query generation for relevance prediction. *arXiv preprint arXiv:2305.11944*.

Derek Chen, Celine Lee, Yunan Lu, Domenic Rosati, and Zhou Yu. 2023. Mixture of soft prompts for controllable data generation. *arXiv preprint arXiv:2303.01580*.

Yan Chen, Shujian Liu, Zheng Liu, Weiyi Sun, Linas Baltrunas, and Benjamin Schroeder. 2022. Wands: Dataset for product search relevance assessment. In *Proceedings of the 44th European Conference on Information Retrieval*, pages 128–141.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B Hall, and Ming-Wei Chang. 2022. Promptagator: few-shot dense retrieval from 8 examples (2022). *arXiv preprint arXiv:2209.11755*.

Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. Climate-fever: A dataset for verification of real-world climate claims. *arXiv preprint arXiv:2012.00614*.

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

Jiahui Gao, Renjie Pi, LIN Yong, Hang Xu, Jiacheng Ye, Zhiyong Wu, WEIZHONG ZHANG, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. 2022. Self-guided noise-free data generation for efficient zero-shot learning. In *The Eleventh International Conference on Learning Representations*.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part I 43*, pages 146–160. Springer.

Himanshu Gupta, Kevin Scaria, Ujjwala Anantheswaran, Shreyas Verma, Mihir Parmar, Saurabh Arjun Sawant, Swaroop Mishra, and Chitta Baral. 2023. Targen: Targeted data generation with large language models. *arXiv preprint arXiv:2310.17876*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. Dbpedia-entity v2: a test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1265–1268.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942.

Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating training data with language models: Towards zero-shot language understanding. *Advances in Neural Information Processing Systems*, 35:462–477.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016*, volume 1773 of *CEUR Workshop Proceedings*.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2021. Large dual encoders are generalizable retrievers.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.

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

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2023. Large language models can accurately predict searcher preferences. *arXiv preprint arXiv:2309.10621*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. Trec-covid: constructing a pandemic information retrieval test collection. In *ACM SIGIR Forum*, volume 54, pages 1–12. ACM New York, NY, USA.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023a. Large language model as attributed training data generator: A tale of diversity and bias. *arXiv preprint arXiv:2306.15895*.

Yue Yu, Yuchen Zhuang, Rongzhi Zhang, Yu Meng, Jiaming Shen, and Chao Zhang. 2023b. ReGen: Zero-shot text classification via training data generation with progressive dense retrieval. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11782–11805, Toronto, Canada. Association for Computational Linguistics.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. Miracl: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Berdersky. 2023. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. *arXiv preprint arXiv:2310.14122*.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. Rankt5: Fine-tuning t5 for text ranking with ranking losses. *arXiv preprint arXiv:2210.10634*.

A Appendix

A.1 Target Task Source

We aim to cover a balanced mixture of tasks covering diverse task categories, domains, and number of relevance labels. Since some of the datasets used in BEIR are constantly updated, to maintain reproducibility and consistency, we download the datasets provided by BEIR authors hosted on TFDS[9]. Although, BEIR has a total of 18 datasets, with the exception of three categories Tweet-Retrieval, News Retrieval, Duplicate-Question Retrieval and Citation Prediction, we include at least one dataset from the other five task categories. Due to proprietary issues, datasets for Tweet-Retrieval and News Retrieval were absent from the TFDS library. We decided to exclude Duplicate-Question Retrieval as the task focused on query-query similarity which means there is no document context as such to generate the query from. Hence, for this work, we decided to focus on tasks having a provided document context.

MS-MARCO released by Nguyen et al. (2016) is constructed from Bing search logs comprising of general purpose web query-document pairs. It comprises of 530,000 queries and 8 million passages, covering two relevance labels, namely, relevant and irrelevant.

TREC-COVID (Voorhees et al., 2021) consists of scientific query-document pairs based on the COVID-19 pandemic. It comprises of 171,332 passages, with a test set of 35,480 examples across three relevance labels, namely, relevant (2.0), somewhat relevant (1.0) and irrelevant (0.0). The BEIR task category for this dataset is Bio-medical IR.

FIQA-2018 (Maia et al., 2018) (Task-2) is opinion-based question answering (QA) task. BEIR mines 57,638 financial articles as the document corpus, with a test set comprising of 1,706 relevant query-document examples. The BEIR task category for this dataset is open-domain QA.

Touché20 (Bondarenko et al., 2020) (Task 1) is a conversational argument retrieval task, with 382,545 documents constructed from the conclusion and premise of arguments. The test set comprises of 2,099 query-document examples, covering three labels, namely, relevant (2.0), somewhat relevant (1.0) and irrelevant (0.0). The BEIR task category for this dataset is Argument Retrieval.

DBPedia (Hasibi et al., 2017) is an entity retrieval dataset, with 4,635,922 DBPedia English articles. The test set has 40,724 query-document examples, covering three labels, namely, relevant (2.0), somewhat relevant (1.0) and irrelevant (0.0). The BEIR task category is Entity Retrieval.

FEVER (Thorne et al., 2018) is a fact-verification task, with 123,142 sentence claims as queries and 5,416,568 Wikipedia abstracts as the document corpus. The test set has 1,499 relevant query-document examples. The BEIR task category for this dataset is Fact Checking.

Climate-FEVER (Diggelmann et al., 2020) is also a fact-verification task, with 5,416,593 Wikipedia abstracts, but with 1,535 real-world climate claims as queries. The test set has 1,344 relevant query-document examples. The BEIR task category for this dataset is Fact Checking.

MIRACL (Zhang et al., 2023) is a multilingual information retrieval dataset covering 18 languages, based on Wikipedia with human-annotated query-passage relevance judgements. We experiment with only the English and Hindi portion of the dataset, with the English dataset comprising of 32M passages and Hindi 506k passages. As before, we randomly sample 50,000 passages for query generation. Since this is an ongoing challenge, gold labels for the test set are not publicly released. Therefore, we use the validation set as our test set, which comprises of 8,350 relevance judgements for English and 3,494 for Hindi.

ESCI comprises of 2.6 million human-labeled query-product relevance judgements, derived from Amazon Search. The query-product pairs are rated using four relevance labels: *exact* when the product exactly matches the query specifications, *substitute* when the product does not match exactly all requirements but could be used as a valid substitute, *complement* when the product could be used in combination with the requested product, and *irrelevant*. ESCI is a multilingual dataset, including query-products for English, Spanish, and Japanese. We use the English portion of the training data for our QGen experiments.

WANDS is also a shopping relevance dataset, but derived from Wayfair Search, comprising of

[9] https://www.tensorflow.org/datasets/catalog/beir

233,448 human-annotated query-product relevance judgements, with 42,994 unique products. The relevance labels are three-way, namely, *exact-match* when the product exact satisfies the query requirements, *partial-match* when the product is relevant to the query with respect to the main entity but differs in the exact modifiers requested, and *irrelevant*. We use the entire dataset as our test split, and all of the 42k products for query generation.

A.2 Model

QGen Model For all tasks, we select 10 query-document examples from the MS-MARCO to curate the prompt (more details on prompt construction in subsection 3.2 and subsection 3.3). We use temperature sampling to generate the outputs, with temperature of 0.6 and beam size of 2. We report the number of queries generated for each step of the QGen process for all models in Table 10 (GENERATE-RELEVANT-ONLY), Table 11 (LABEL-CONDITIONED), Table 9 (GENERATE-PAIRWISE), and Table 12 (ITERATIVE-PAIRWISE).

Downstream Model As described in subsection 3.1, we train a mt5-XXL downstream model on the generated data for each task. We train the mt5-XXL model to run for at least 1 epoch of the training data, using a learning rate of $5e-05$, batch size of 64, input sequence length of 512, and Adafactor optimzer. Note that we use a smaller model for our downstream task as compared to our query generation model, as a smaller downstream model is more compute and time efficient, which is extremely important for real-world downstream application, where the inference needs to be run several times and often on large test sets. The downstream model takes 2-3 hours to complete training, where we use 128 TPU chips. For each QGen setting and task, we need to run three models , namely, query generation, query filtration and, downstream model. Given that we run these models for 4 QGen settings across 9 IR tasks, we ony report results for single run.

A.3 QGen for Fine-grained Relevance Prediction Setup

Query Filtration As described in subsection 2.3, we prompt the model to generate the relevance label, and remove those queries whose predicted label does not match the label using which it was generated. Earlier, we had use the LLM's scoring mode for filtration, in this setting, we directly make the model generate the relevance label for a faster inference. For scoring across four labels, the LLM would have to make four calls for each input, which is expensive when the generated queries are O(100k).

A.4 Invalid Prompt Outputs

LLMs often generate invalid or useless outputs (Qin et al., 2023). For the QGen experiments on BEIR datasets, where queries were generated for binary relevance labels, we find from Table 9, Table 10, Table 11, and Table 12 that the percentage of such invalid queries generated is nearly 100% for GENERATE-RELEVANT-ONLY and LABEL-CONDITIONED, while for GENERATE-PAIRWISE and ITERATIVE-PAIRWISE it drops to 85% and 87% respectively. Some common reasons for such invalid outputs are lack of 'query' or 'label' prefix which causes incorrect parsing. For QGen approaches where only one query output is expected (i.e. GENERATE-RELEVANT-ONLY and LABEL-CONDITIONED) such issues are not that common but for the other two approaches where two queries need to be generated such errors are observed more. We observe this issue more for GENERATE-LABELS approach used in the fine-grained relevance prediction case study, where the model is expected to generate four queries for four labels. In particular, we find 46% of generated queries are invalid or useless which dramatically reduces the synthetic data pool. We find that most of these are deemed invalid because the LLM starts generating garbage and does not adhere to the task instruction, nearly all of the 46% invalid queries are outputs where the LLM generates information about a new product, its title and description which is typically provided as input.

Generate-Relevant-Only
Given a product generate a query that exactly matches the product specifications: product: MOUNTAINTOP 40L Hiking Backpacks with Rain Cover for Women Men MOUNTAINTOP 40L Hiking Backpack, A must have for hiking, camping, traveling and cycling, helps you reach the most epic views the world has to offer query: mountaintop hiking pack product: `new product`
Label-Conditioned
Given a product and desired relevance label generate a query that is appropriate for that relevance label. The four relevance labels are 'Exact' which means that the item is relevant for the query, and satisfies all the query specifications . 'Substitute' means that the item is somewhat relevant, i.e., it fails to fulfill some aspects of the query but the item can be used as a functional substitute. 'Complement' means that the item does not fulfill the query, but could be used in combination with an exact item. 'Irrelevant' means that the item is irrelevant, or it fails to fulfill a central aspect of the query. Some examples are: product: MOUNTAINTOP 40L Hiking Backpacks with Rain Cover for Women Men MOUNTAINTOP 40L Hiking Backpack, A must have for hiking, camping, traveling and cycling, helps you reach the most epic views the world has to offer label: Exact query: mountaintop hiking pack product: Office Chair Ergonomic Cheap Desk Chair Mesh Computer Chair Lumbar Support Modern Executive Adjustable Stool Rolling Swivel Chair for Back Pain, Black label: Substitute query: office chair without wheels or lift product: `new product` label: `Exact/Substitute/Complement/Irrelevant`
Generate-Pairwise
Given a product and a desired relevance label, the task is to generate two unique query for each relevance label. The four relevance labels are 'Exact' which means that the item is relevant for the query, and satisfies all the query specifications . 'Substitute' means that the item is somewhat relevant, i.e., it fails to fulfill some aspects of the query but the item can be used as a functional substitute. 'Complement' means that the item does not fulfill the query, but could be used in combination with an exact item. 'Irrelevant' means that the item is irrelevant, or it fails to fulfill a central aspect of the query. product: MOUNTAINTOP 40L Hiking Backpacks with Rain Cover for Women Men MOUNTAINTOP 40L Hiking Backpack, A must have for hiking, camping, traveling and cycling, helps you reach the most epic views the world has to offer task: generate query1 for Exact and query2 for Substitute query1: mountaintop hiking pack query2: osprey jet 12 product: Office Chair Ergonomic Cheap Desk Chair Mesh Computer Chair Lumbar Support Modern Executive Adjustable Stool Rolling Swivel Chair for Back Pain, Black task: generate query1 for Substitute and query2 for Complement query1: office chair without wheels or lift query2: ergonimic foot stool product: `new product` task: generate query1 for `label-1` and query2 for `label-2`
Generate-AllLabels
Given a product and a desired relevance label, the task is to generate a unique query for each relevance label. The four relevance labels are 'Exact' which means that the item is relevant for the query, and satisfies all the query specifications . 'Substitute' means that the item is somewhat relevant, i.e., it fails to fulfill some aspects of the query but the item can be used as a functional substitute. 'Complement' means that the item does not fulfill the query, but could be used in combination with an exact item. 'Irrelevant' means that the item is irrelevant, or it fails to fulfill a central aspect of the query. product: MOUNTAINTOP 40L Hiking Backpacks with Rain Cover for Women Men MOUNTAINTOP 40L Hiking Backpack, A must have for hiking, camping, traveling and cycling, helps you reach the most epic views the world has to offer Label: Exact Query: mountaintop hiking pack Label: Substitute Query: osprey jet 12 Label: Complement Query: waterproof shoes hiking Label: Irrelevant Query: mountaintop whitlow product: `new product`

Table 6: Prompt formats for the different QGen models used in section 5.

Generate-Relevant-Only
Given a passage from a web page, generate a search query for which the passage can be a perfect answer. passage: Premature Ventricular Contractions (PVCs, PVC) Medical Definition of Cardiac stress testing, exercise. Cardiac stress testing, exercise: The exercise cardiac stress testing (EST) is the most widely used cardiac (heart) screening test. The patient exercises on a treadmill according to a standardized protocol, with progressive increases in the speed and elevation of the treadmill (typically changing at three-minute intervals). query: what is cardiac testing in medical terms passage: `new passage` query: `relevant query`
Label-Conditioned
Given a passage from a web page and a relevance label, generate a search query appropriate for that relevance level for that passage. If the label is "relevant", the query should be such that the passage can be a perfect answer and if the label is "irrelevant" the query should be such that the passage is not a perfect answer. passage: Premature Ventricular Contractions (PVCs, PVC) Medical Definition of Cardiac stress testing, exercise. Cardiac stress testing, exercise: The exercise cardiac stress testing (EST) is the most widely used cardiac (heart) screening test. The patient exercises on a treadmill according to a standardized protocol, with progressive increases in the speed and elevation of the treadmill (typically changing at three-minute intervals). label: relevant query: what is cardiac testing in medical terms passage: Amazon Customer Service Whatever the issue, you're going to want to get in touch with Amazon's customer service department. The easiest way to contact Amazon's customer service department is by using their toll-free phone number at 1-888-280-4331. label: irrelevant query: amex customer service phone number passage: `new passage` label: `relevant / irrelevant` query: `relevant / irrelevant query`
Generate-Pairwise
Given a passage from a web page, generate a search query for which the passage can be a perfect answer and a search query for which the passage is not a perfect answer. passage: Premature Ventricular Contractions (PVCs, PVC) Medical Definition of Cardiac stress testing, exercise. Cardiac stress testing, exercise: The exercise cardiac stress testing (EST) is the most widely used cardiac (heart) screening test. The patient exercises on a treadmill according to a standardized protocol, with progressive increases in the speed and elevation of the treadmill (typically changing at three-minute intervals). query1: what is cardiac testing in medical terms query2: how soon exercise after heart stent passage: Amazon Customer Service Whatever the issue, you're going to want to get in touch with Amazon's customer service department. The easiest way to contact Amazon's customer service department is by using their toll-free phone number at 1-888-280-4331. query1: what is amazon phone number customer service query2: amex customer service phone number passage: `new passage` query1: `relevant query` query2: `irrelevant query`

Table 7: Prompt formats for the different QGen models used in section 3.

	QGen Model	en	hi
Baseline	Generate-Relevant-Only	77175	72874
	Label-Conditioned	106107	104850
Ours	Generate-Pairwise	181116	170805
	Iterative-Pairwise	77788	77897

Table 8: Number of synthetic query-passage relevance judgements generated on the MIRACL datasets.

Dataset	QGen Inputs		Process QGen Outputs		Training Data					
	Prompt Inputs	**Requested Queries**	**Valid Query Outputs**	**Filtered Query Outputs**	**Train Examples**	**Irrelevant Examples**	**Relevant Examples**	**% Valid Queries**	**% Valid Examples**	**Irrelevant / Relevant**
trec-covid	128821	257642	457072	257204	257204	85385	171819	0.89	0.5	0.5
touche	50001	100002	168370	88295	88295	67495	20800	0.84	0.44	3.24
dbpedia	50001	100002	167317	93655	93655	73587	20068	0.84	0.47	3.67
climate-fever	50001	100002	166364	88872	88872	76488	12384	0.84	0.44	6.18
fiqa	57013	114026	199491	109593	109593	72296	37297	0.87	0.48	1.94
fever	50001	100002	166047	88696	88696	76864	11832	0.83	0.44	6.5

Table 9: We report the data statistics for each step of GENERATE-PAIRWISE model for all datasets. c

Dataset	QGen Inputs		Process QGen Outputs		Training Data					
	Prompt Inputs	**Requested Queries**	**Valid Query Outputs**	**Filtered Query Outputs**	**Train Examples**	**Irrelevant Examples**	**Relevant Examples**	**% Valid Queries**	**% Valid Examples**	**Irrelevant / Relevant**
trec-covid	128821	257642	257497	197995	360465	162470	197995	0.99	0.77	0.82
touche	50001	100002	96003	25598	51196	25598	25598	0.96	0.26	1.0
dbpedia	50001	100002	100001	24064	46198	22134	24064	0.99	0.24	0.92
climate-fever	50001	100002	99978	14171	27208	13037	14171	0.99	0.14	0.92
fiqa	57013	114026	113536	22182	43814	21002	22182	0.99	0.19	0.95
fever	50001	100002	99970	13793	26585	12792	13793	0.99	0.14	0.93

Table 10: We report the data statistics for each step of GENERATE-RELEVANT-ONLY model for all datasets. The % of valid queries refers to the percentage of valid queries after the query generation step while the % of valid examples is after the filtration step.

Dataset	QGen Inputs		Process QGen Outputs		Training Data					
	Prompt Inputs	**Requested Queries**	**Valid Query Outputs**	**Filtered Query Outputs**	**Train Examples**	**Irrelevant Examples**	**Relevant Examples**	**% Valid Queries**	**% Valid Examples**	**Irrelevant / Relevant**
trec-covid	257642	515284	515258	305238	305238	117693	190485	0.99	0.59	0.62
touche	100002	200004	199992	103542	103542	78561	26697	0.99	0.52	2.94
dbpedia	100002	200004	200000	105665	105665	85981	22129	0.99	0.53	3.89
climate-fever	100002	200004	199987	103043	103043	91465	13558	0.99	0.52	6.75
fiqa	114026	228052	228052	118615	118615	71905	47289	0.99	0.52	1.52
fever	100002	200004	199986	102924	102924	91879	12970	0.99	0.51	7.08

Table 11: We report the data statistics for each step of LABEL-CONDITIONED model for all datasets. The % of valid queries refers to the percentage of valid queries after the query generation step while the % of valid examples is after the filtration step.

Dataset	QGen Inputs			Process QGen Outputs	Training Data				
	Prompt Inputs	**Requested Queries**	**Relevant Query Outputs**	**Irrelevant Query Requested**	**Train Outputs**	**Irrelevant Examples**	**% Valid Queries**	**% Valid Examples**	**Irrelevant / Relevant**
trec-covid	128821	257642	197995	404323	287490	485485	0.71	0.94	1.45
touche	50001	100002	25598	50524	39379	64977	0.77	0.32	1.54
dbpedia	50001	100002	24064	47980	41748	65812	0.87	0.33	1.73
climate-fever	50001	100002	14171	28218	28555	42726	1.0	0.21	2.02
fiqa	57013	114026	22182	44018	38749	60931	0.88	0.27	1.75
fever	50001	100002	13793	27485	27877	41670	1.0	0.21	2.02

Table 12: We report the data statistics for each step of ITERATIVE-PAIRWISE model for all datasets. The % of valid queries refers to the percentage of valid queries after the query generation step while the % of valid examples is after the filtration step.

Because political parties control primary elections, party bosses are given substantial power and authority in making decisions that lead to the election of the next president.This is undemocratic.	I would not need to explain what chopsticks are. I assume the position that a set of chopsticks is a superior eating utensil to a fork.
Generate-Relevant-Only	
relevant: why primary elections are undemocratic	why are chopsticks superior to forks
irrelevant: how do superdelegates undermine the democratic party	chop sticks are better than forks
Label-Conditioned	
relevant: why are political parties undemocratic	why is chopsticks better than fork?
irrelevant: why are political parties bad	what are the best eating utensils
Generate-Pairwise	
relevant: why are primaries undemocratic	are chopsticks superior to a fork
irrelevant: why is voting not democratic	why chopsticks are better than knife
Iterative-Pairwise	
relevant: why primary elections are undemocratic	why are chopsticks superior to forks
irrelevant: why primary elections	what is the best utensil for eating

Table 13: We compare the generated queries across different QGen models for the Touché dataset. The first row denotes the passage snippets used for the query generation. Note for the GENERATE-RELEVANT-ONLY the irrelevant queries are not specifically generated rather the generated relevant queries are paired with retrieved documents.

The Painter and The Buyer is a 1565 pen and ink on brown paper painting by Flemish artist Pieter Bruegel the Elder. The alternative title is The Artist and The Connoisseur.The painter is thought to be a self-portrait of Pieter Bruegel the Elder.	Mateuți is a commune in Rezina District, Moldova. It is composed of a single village , Mateuți .
Generate-Relevant-Only	
relevant: what is the painter and the buyer painting	what is mateuti
irrelevant: who is drew halfmann	what is dancu district in moldova
Label-Conditioned	
relevant: who is painter in painter and buyer	what is mateuti
irrelevant: what is the difference between a painting and a picture	how to read vitamin labels
Generate-Pairwise	
relevant: is painter and the buyer a 1565 pen and ink on brown paper painting by Flemish artist Pieter Bruegel the Elder	what is mateuti
irrelevant: what is the name of the painter	what is the most common name in the world
Iterative-Pairwise	
relevant: what is the painter and the buyer painting	what is mateuti
irrelevant: what is the painter and the buyer about	what is oxegen

Table 14: We compare the generated queries across different QGen models for the FEVER dataset. The first row denotes the passage used for the query generation. Note for the GENERATE-RELEVANT-ONLY the **relevant** queries are not specifically generated rather the generated relevant queries are paired with retrieved documents.

rangeworthy 67 ” genuine leather tuxedo arm loveseat	stainless steel modern pull 3 3/4 ” center to centerfinger pull
Generate-Relevant-Only	
exact: leather sofa for living room	stainless steel modern pull
not-exact: 67 ” leather loveseat	stainless steel pull
Label-Conditioned	
exact: leather couch	stainless steel pull
substitute: -	-
complement: leather sectional for small space	best kitchen cabinet hardware
irrelevant: baby girl shoes	baby crib
Generate-Pairwise	
exact: leather sectional furniture	black stainless steel door pulls
substitute: red leather lounge chair	stainless steel door knob
complement: leather recliner for senior	-
irrelevant: leather chair for gaming	4 in 1 baby car seat
Generate-AllLabels	
exact: tuxedo arm loveseat	stainless steel modern pull 3 3/4 ” center to centerfinger pull
substitute: leather couch 70 inches	black modern drawer pulls
complement: -	-
irrelevant: -	black modern furniture

Table 15: We compare the generated queries for different QGEn models for the WANDS dataset. Note, exemplars from ESCI are used for the query generation. The first row denotes the WANDS product used for the query generation.