

Beyond Manual Prompts: In-Context Learning for LLM Query Expansion for information retrieval via Auto-Generated Pseudo-Relevance Datasets

Anonymous ACL submission

Abstract

Query expansion (QE) enhances information retrieval (IR) by addressing vocabulary gaps between queries and documents. While large language models (LLMs) enable generative QE through in-context learning with few examples, existing methods rely on manual prompts or static datasets, limiting domain adaptability and systematic evaluation of few-shot selection strategies. We propose an automated framework to construct domain-adaptive QE candidate datasets¹ without human annotation. Leveraging an unlabeled target-domain corpus and a BM25-then-MonoT5 retrieval pipeline, our method extracts pseudo-relevant passages from seed queries, transforming them into few-shot exemplar candidates. We evaluate four selection strategies for LLM demonstrations: static, random, clustering-based diversity, and embedding-based similarity. Experiments across web search (TREC 2019, 2020 DL Track), financial (FiQA), and open-domain entity queries (DBpedia) using Qwen-2.5-7B-Instruct show that LLM-generated expansions largely improve BM25 retrieval performance. Our framework provides a scalable, domain-adaptive solution for in-context query expansion with LLMs—serving as both a reproducible benchmark for evaluation and a practical tool for real-world deployment, while enabling further research on in-context learning few-shot selection from large candidate pools.

1 Introduction

Information retrieval (IR) systems often suffer when user queries use vocabulary or phrasing that differs from relevant documents. Query expansion (QE) techniques (Wang et al., 2023) have long been used to address this by adding alternate terms or reformulating queries to better match the language of relevant documents. Traditional QE methods, such

¹We make the generated candidate datasets public available at <https://huggingface.co/XXXXX>

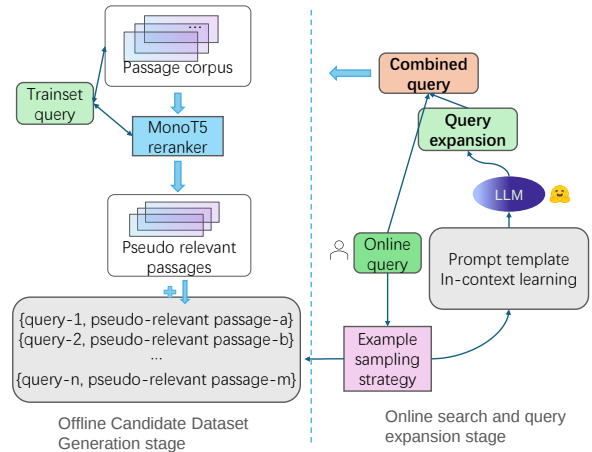


Figure 1: Overview of our automated pipeline for constructing domain-adaptive few-shot query expansion datasets and evaluating in-context learning strategies.

as Pseudo-Relevance Feedback (PRF) (Cao et al., 2008) like Rocchio (Miao et al., 2012; Liu, 2022) or RM3 (Abdul-Jaleel et al., 2004), assume an initial retrieval and then expand the query with terms from top-ranked documents. While often effective, these methods are tightly coupled to the quality of the first-stage retrieval. That is, they require an initial search to be run before expansion can be performed, making them less suitable for scenarios where immediate query enhancement is needed or when the initial retrieval is poor. Moreover, they do not leverage external linguistic knowledge beyond the corpus, limiting their ability to introduce semantically rich variations.

Large Language Models (LLMs) (Kalyan, 2024) provide a new paradigm for QE by generating semantically related queries or text using their vast knowledge. Recent studies have shown that LLMs can produce expansions that improve recall and downstream retrieval performance. In particular, in-context learning with LLMs allows us to prompt an LLM with a few example query expansions (without any fine-tuning) (Dong et al., 2024) and have it

generate an expanded query for a new user query. This few-shot prompting approach is attractive for IR tasks since it avoids the need for retraining models for each new domain or query type. However, a key challenge is determining which examples to include in the prompt. [Brown et al. \(2020\)](#) showed that performance can vary widely depending on which examples are chosen and in what order, even for GPT-3. Recent research has attempted to make example selection more systematic. [Wang et al. \(2024\)](#) learn a dense retriever to pick in-context examples by training on feedback from the LLM, demonstrating improved performance on multiple tasks by retrieving examples with similar “patterns” to the query. In the IR community, however, there is little work in find optimal prompt examples for query expansion. In our benchmark, we experiment with heuristic selection strategies: (a) semantic similarity (which is an unsupervised proxy for relevance), and (b) diversity via clustering (to cover different query patterns). These represent intuitive baselines for automated example retrieval, bracketing the space between always using the same examples (static) and picking randomly. To our knowledge, our work is among the first to explicitly benchmark different few-shot selection strategies in the context of query expansion for IR. Besides, we simulate an common scenario when we not have labels of query-relevant passage pairs for new datasets. By providing a concrete dataset and evaluation, we enable future research to plug in learned retrievers or more sophisticated selection criteria and measure their impact on retrieval performance.

Our contributions include:

- We propose a fully automated and scalable pipeline for constructing pseudo-labeled query expansion datasets from unlabeled corpora using BM25 and MonoT5, without requiring manual annotation.
- We release a benchmark covering three domains (TREC DL19/20, FiQA, DBPedia), enabling systematic study of in-context query expansion with large exemplar pools.
- We compare four exemplar selection strategies—static, random, nearest-neighbor, and clustering—and find that the proposed in-context learning strategies without manual examples can still improve QE performances.

2 Related Work

Query Expansion in Information Retrieval.

Query expansion has been studied for decades as a method to improve search recall. Early approaches include manual expansion using thesauri and semantic resources, as well as automatic expansion using pseudo-relevance feedback (PRF) ([Climb and Gaussier, 2013](#)). In PRF, an initial search is performed and the top-ranked documents are assumed to be relevant; terms from these documents are then added to the query (often with weighting) to perform a second, expanded search. Classic methods like Rocchio’s algorithm ([Miao et al., 2012](#); [Liu, 2022](#)) and the probabilistic Relevance Model (RM3) demonstrated the effectiveness of PRF for both keyword-based and probabilistic IR models. However, PRF can drift the query topic if the initially retrieved documents are not truly relevant, and it typically only adds individual terms without understanding context. Neural approaches to expansion have emerged in recent years. One line of work is to use sequence-to-sequence models to generate expansions or related queries. For example, the *doc2query* method proposed by [Nogueira et al. \(2019\)](#) uses a neural model to generate probable queries that a given document could answer. In practice, *doc2query* (and its T5-based variant *doc2query-T5*) was used to expand each document in the corpus with several pseudo-queries, which are then indexed to improve recall for original user queries. This is an offline expansion of the document collection, complementary to expanding the query itself. Our approach, in contrast, focuses on online query expansion: we expand the user’s query at query time. We similarly leverage a sequence-to-sequence model (an LLM) to produce the expansion, but condition it on retrieved content for grounding, akin to pseudo-relevance feedback but with generative re-writing. Another line of neural expansion research directly uses generative models to expand queries at runtime. Recent work on *Generative Relevance Feedback (GRF)* by [Mackie et al. \(2023\)](#) proposes to generate long-form text (e.g., an imagined relevant document or essay) from the query using an LLM, and then derive expansion terms from that text. This approach does not rely on actual retrieved documents, instead leveraging the language model’s inherent knowledge to predict relevant content. Experiments have shown GRF can outperform traditional PRF (RM3) on diverse retrieval tasks, improving nDCG@10 by a substantial

margin . Similarly, Jagerman et al. (2023) explored prompting GPT-3 style LLMs for query expansion and found that certain prompt styles, especially chain-of-thought prompting, yielded more effective expansions. Wang et al. (2023) propose few-shot learning with LLMs for query expansion. These works illustrate the promise of LLMs in generating useful expansion text beyond simple term addition. On the other hand, a challenge noted in subsequent studies is that unconstrained LLM generation can introduce irrelevant or hallucinated content that might hurt retrieval . For instance, an LLM might introduce facts or entities not present in the corpus or deviate from the query intent if not guided properly. To mitigate this, researchers have proposed hybrid approaches that steer LLM expansions using the corpus itself. One such approach is *Corpus-Steered Query Expansion (CSQE)* , which uses an initial BM25 retrieval (Robertson et al., 2009) to get some documents, then asks an LLM to extract or emphasize information from those documents when generating the expansion. This grounds the expansion in actual content known to exist in the corpus, reducing hallucination and making the expansion more effective for retrieval . Our method is closely aligned with this idea: we explicitly use top-retrieved passages (via BM25 + reranker) as the basis for expansions. In our case, we even incorporate the passage text directly as the expanded query example (optionally processed by an LLM for brevity), ensuring that the expansion consists of real-world terms and phrases from the target domain.

In-Context Learning and Example Selection.

The ability of LLMs to perform tasks via in-context learning (ICL) has drawn parallels to information retrieval itself . In ICL, a few input-output examples are provided in the prompt, and the model is expected to produce the output for a new input without parameter updates . This mechanism can be seen as the model “retrieving” patterns from the examples to apply to the new query, analogous to how a nearest-neighbor classifier might use similar past cases . Therefore, selecting good examples is crucial. Brown et al. (2020) showed that performance can vary widely depending on which examples are chosen and in what order, even for GPT-3. Recent research has attempted to make example selection more systematic. Wang et al. (2024) learn a dense retriever to pick in-context examples by training on feedback from the LLM, demonstrating improved

performance on multiple tasks by retrieving examples with similar “patterns” to the query. In the IR community, there is a growing interest in applying retrieval algorithms to find optimal prompt examples. In our benchmark, we compare heuristic selection strategies spanning from static and random choices to semantic similarity-based retrieval and cluster-based diversity. By evaluating these strategies, we highlight the importance of intelligent example selection for in-context learning in IR.

3 Methodology

Our goal is to construct a dataset of query expansion examples in an automated fashion for any given domain, and then use that dataset to evaluate few-shot query expansion with LLMs. The overall pipeline is shown in Fig. 1, which consists of two stages: (1) *Offline Candidate Dataset Generation stage*, and (2) *Online search and query expansion stage*. Below, we describe each stage in detail, in and Section 3.1 and Section 3.2 respectively.

3.1 Automatic Expansion Example Generation

Given an unlabeled document corpus for the target domain, we first generate a pool of pseudo query-expansion examples. Each example will consist of a query and an expanded version of that query (in the form of a passage or detailed text) that is likely to be relevant to the query. We achieve this by leveraging the document corpus itself as a source of query expansions, in a manner inspired by pseudo-relevance feedback and query generation techniques (Wang et al., 2022). The steps are described below.

Seed Query Selection. We assume access to a set of seed queries related to the domain. In many cases, these can be obtained from existing data: for example, the queries in the training set. However, we do not require any relevance labels for these queries – they are used only to retrieve content. In our experiments, we use the training queries from the respective datasets (e.g., MS MARCO training queries, FiQA training questions) as seeds.

Initial Retrieval with BM25. For each seed query, we perform first-stage retrieval on the domain’s corpus using BM25, a strong lexical ranking baseline. BM25 efficiently returns a set of top candidate documents or passages that contain keywords overlapping with the query. We

denote the top N retrieved texts for query q as $D(q) = d_1, d_2, \dots, d_N$, sorted by BM25 score. In our implementation, we used the Anserini IR toolkit with its default BM25 parameters ($k_1 = 0.9$, $b = 0.4$) to index each corpus and retrieve top $N = 100$ results for each query.

Reranking with MonoT5. The initial BM25 results may contain some irrelevant items. To improve precision, we rerank the top candidates using a neural reranker. We employ MonoT5 (Nogueira et al., 2020), a sequence-to-sequence reranking model. MonoT5 takes a query and a candidate passage as input and outputs a relevance score (often accomplished by having the model generate a token like “true” or “false” to indicate relevance). In practice, MonoT5 has shown excellent reranking performance and robust zero-shot transfer to other datasets. We use a MonoT5 3B model² fine-tuned on the MSMARCO passage ranking task to score the top-100 BM25 list for each query. The highest-scoring (top-1) passage $d_q^* = \arg \max_{d \in D(q)} \text{MonoT5Score}(q, d)$ is taken to be a pseudo-relevant passage for the query q , where $D(q)$ is the BM25 top list. This reranker can significantly improve the chances that d_q^* is actually relevant to q , compared to using BM25’s top result alone.

Dataset Assembly. After the above steps, we have, for each seed query q , a tuple (q, p_q) where p_q is the expanded passage (i.e. d_q^* in this case). We add this as one example in our expansion candidate dataset. Before finalizing, we apply some basic cleaning: removing any excessive whitespace or control characters. We also ensure that none of our seed queries coincide with the evaluation (test) queries, to avoid any trivial overlap. In practice, if a seed query set is the training set of a benchmark and the test queries are separate, this is naturally satisfied. The output of the generation stage is a Jsonl file containing lines. Each line contains the query id, original query and the expansion (p_q). In total, our MS MARCO expansion dataset contains 100K examples (we used a random subset of MS MARCO training queries for seed, up to 100,000 for manageability), and the FiQA expansion dataset is smaller (we used the 5500 finance questions available from FiQA training set), resulting 5500 examples. For the DBPedia-Entity

dataset, it only contains another dev set except test set. So the dev set is used to generate candidates, resulting 64 examples, which is smaller. The methodology is scalable; the size can be adjusted as needed or even include all available queries. This automated construction fulfills our aim of creating a domain-adaptive dataset: if the corpus is from a new domain, the content of expansions will reflect that domain’s jargon and contexts. No manual writing of expansions is required. These datasets can also serve as the few-shot ICL sample pool, and used for sampling strategy research for ICL.

3.2 Few-Shot Query Expansion with LLMs

Given the automatically constructed example pool, our second stage generates an *LLM-expanded query* for every incoming user query q^{test} . The core idea is to supply exactly four *query* $\rightarrow$ *expansion* demonstrations to the **Qwen-2.5-7B-Instruct** model and let it complete the next turn of the conversation.

3.2.1 Prompt template

We use a conversational prompt template with a system message to guide the LLM’s generation style. This format is compatible with many instruction-following LLMs.

The prompt is composed of:

- A single system message to instruct the LLM to write expansions in fluent, domain-relevant language.
- Four User–Assistant message pairs (q_i, p_i) as demonstrations.
- A final User message containing the test query q^{test} .

In full, the dialogue structure reads:

```
<system> "You are an assistant that generates
detailed passages to answer search queries.
Your responses should be informative, directly
address the query, and provide comprehensive
explanations or solutions."
<user> q1
<assistant> p1
<user> q2
<assistant> p2
<user> q3
<assistant> p3
<user> q4
<assistant> p4
<user> Query: qtest
Please write a passage (60–100 words) that
answers it.
```

²<https://huggingface.co/castorini/monot5-3b-msmarco>

This format is dynamically populated with example pairs selected by one of our strategies (see §3.2.2). The assistant is expected to output a concise and informative passage that reflects the structure and tone of prior examples. We use a YAML-based pipeline to construct this prompt automatically for each query at inference time.

3.2.2 How to pick the four demonstrations?

We explore four *training-free* policies, listed below with additional implementation details beyond the short description in § 3.1:

1. **Static**: choose the very first four examples of the example candidate data produced by Stage 1.
2. **Random**: draw four distinct examples uniformly from the candidate example dataset, using a fixed seed so experiments are reproducible.
3. **Embed (Similarity-based)**: We precompute embeddings for each candidate example by applying a Contriever-based encoder³, ported to the SentenceTransformers framework (Reimers and Gurevych, 2019), to the concatenated query and passage text. At test time, we encode the new query q^{test} and retrieve the top- k most similar examples by cosine similarity in embedding space. This unsupervised nearest-neighbor strategy enables semantic alignment between the test query and selected exemplars, without requiring any task-specific retriever training.
4. **Cluster (Diversity-based)**: We apply k -means clustering to the same query–passage embeddings used above, partitioning the candidate pool into k semantic groups. From each cluster, we select the *medoid*—the real example closest to the centroid in Euclidean space—as a representative. This yields a fixed but diverse subset of k exemplars that encourages broader topic coverage in the prompt.

The full procedures of Embed and Cluster are provided in Appendix A, Algorithms 1 and 2.

3.2.3 Using the Expanded Query for Retrieval

Once the expansion passage p^{exp} is produced, we *concatenate* it to the original user query q^{test} to

³nishimoto/contriever-sentence-transformer

build the retrieval string following (Wang et al., 2023):

$$q^{\text{new}} = \underbrace{q^{\text{test}}}_{\text{repeat}} \times 5 \parallel p^{\text{exp}}$$

where $\parallel$ denotes string concatenation and we repeat q^{test} five times. The concatenated string is then fed to the retrieval system.

4 Experimental Setup

4.1 Tasks and corpora

Benchmarks. We evaluate on four retrieval test sets drawn from three public corpora:

- TREC DL19 (Craswell et al.): 43 topics with graded relevance judgement.
- TREC DL20 (Craswell et al., 2021): 54 topics with graded relevance judgement. We use the official “passage” task; both DL19 and DL20 share the same 8.8M msmarco passage corpus.
- FiQA-2018 (Thakur et al.): 648 consumer-finance queries (binary qrels) over 57 k documents.
- DBPedia-Entity (Thakur et al.): 400 entity-centric queries over 4.9 M Wikipedia abstracts.

Dataset	#Test Queries	Corpus size	Pool size
DL19	43	8.8 M	100000
DL20	54		
FiQA	648	57 k	5500
DBPedia	400	4.9 M	64

Table 1: Evaluation sets and statistics. “Pool size” = number of candidate query–passage pairs harvested by MonoT5 and subsequently available for demonstrations sampling. The number in **bold** represents the number of generated pseudo-labeling example candidates in datasets.

4.2 Indexing and first-stage retrieval

All corpora are indexed with Anserini/Lucene JAVA implementation (Yang et al., 2018). BM25 with default parameters ($k_1=0.9$, $b=0.4$) retrieves the top 1000 passages for every test query. TREC DL19 and DL20 passage ranking tasks share the same MS MARCO corpus and BM25 index.

Pseudo-label harvesting. BM25 top-100 lists are reranked with MonoT5-3B; we keep the highest-scoring passage per query, yielding the demonstration pools summarized in Table 1: 100 k pairs for MS MARCO (shared by DL19 and DL20), 5.5 k for FiQA (from training set), and 64 for DBPedia-Entity (from the only available dev set). These pools serve both as training material and as the source from which few-shot exemplars are drawn in Stage 2.

4.3 LLM expansion model

All expansions are generated by **Qwen-2.5-7B-Instruct**⁴ (Bai et al., 2023) model with 7B parameters without any fine-tuning. We prompt the model with a chat dialogue containing four (this means $k=4$ for few-shot sampling and k -means) (q_i, p_i) demonstrations (each passage truncated to 60 words) plus the test query. They are fit within a 1024-token context window. Decoding uses 4-beam search, `max_new_tokens=64`, `repetition_penalty=1.1`, and `no_repeat_ngram_size=2`. Experiments are down on a Nvidia A100 GPU. The generation requires only several minutes with 648 queries as an example.

4.4 Evaluation metrics and baselines

Metrics. Effectiveness is measured with `nDCG@10`, `P@10`, `MRR@10`, and `Recall@1000` computed by `trec_eval`.

Baselines.

- **BM25:** Original query without any expansion. Serves as the lexical retrieval baseline.
- **BM25+Rocchio:** Classic pseudo-relevance feedback (Liu, 2022) using Anserini’s standard Rocchio implementation. We use the top 10 retrieved documents, with $\alpha = 1.0$ and $\beta = 0.75$, and interpolate the original query with the top 10 feedback terms.
- **ChatExp-0:** Zero-shot prompting using the same LLM prompt template but with no in-context demonstrations. This isolates the effect of including examples.
- **ChatExp-Fixed:** Few-shot prompting with the same four exemplars as used in Wang et al. (2023), held constant across all test queries.

⁴<https://huggingface.co/Qwen/Qwen2.5-7B-Instruct>

Our variants. All systems below share the same candidate pool and Lucene index as the baselines, isolating the impact of example selection.

- **ChatExp-Static:** first four examples in the shuffled pool.
- **ChatExp-Random:** four examples drawn uniformly at random for each query (seed 42).
- **ChatExp-NN:** four nearest neighbours selected by Contriever embedding similarity (“Embed” in §3.2).
- **ChatExp-Cluster:** four cluster-medoid exemplars chosen by k -means ($k=4$) for maximum topical diversity.

All LLM-based runs use the identical prompt template and Qwen-2.5-7B-Instruct generation settings.

5 Results and Analysis

5.1 How strong are the lexical baselines?

The first two rows of Table 2 and 3 establish a lexical upper bound. Compared with vanilla **BM25**, **BM25+Rocchio** improves recall (`R@1000`) by +5–6 points on the MS MARCO tracks but degrades `nDCG@10` on FiQA (−3.9) and DBPedia (−1.0). This confirms classic observations: Rocchio tends to help when the corpus is large and relevance density low (MS MARCO), but can inject noise in niche domains with short queries (FiQA) or when judged documents are sparse (DBPedia). Any LLM-based method must therefore beat *both* baselines to be considered broadly useful.

5.2 Effect of zero-shot prompting

ChatExp-0 (row 3) already delivers sizeable gains over both lexical runs on all four datasets. The effect is largest on DL’19 (+9.9 `nDCG@10`) and smallest on FiQA (+1.4). We attribute this to the long-form passages generated by Qwen injecting rich synonymy, which is particularly helpful for web queries with verbose relevance descriptions.

5.3 Value of fixed exemplars

ChatExp-Fixed mirrors the Query2Doc (Wang et al., 2023) prompt. Using those four hand-curated MS MARCO examples transfers reasonably well to FiQA (+0.9 `nDCG`) and DBPedia (+0.1), suggesting that *domain mismatch hurts less than example quality*. However, the fixed set fails to outperform

Method	NDCG@10	P@10	MRR	R@1000	NDCG@10	P@10	MRR	R@1000
TREC DL 19					TREC DL 20			
BM25	50.58	61.86	82.45	73.89	47.96	53.89	82.69	72.28
BM25+Rocchio	52.75	66.28	79.80	78.82	49.10	57.22	80.59	77.16
ChatExp-0	60.47	70.70	89.82	79.54	53.77	61.30	86.72	75.61
ChatExp-Fixed	60.69	73.02	88.53	80.14	53.63	60.93	84.78	77.06
ChatExp-Static	59.94	72.33	88.84	79.55	54.57	62.04	86.57	78.39
ChatExp-Random	60.14	72.09	89.03	80.49	52.12	59.44	83.32	76.98
ChatExp-Cluster	60.58	72.33	90.39	79.25	53.07	60.19	87.31	76.07
ChatExp-NN	61.73	73.72	89.06	79.00	54.88	63.70	85.21	77.85

Table 2: Retrieval performance (%) on TREC DL’19 and DL’20. “R@” stands for Recall@.

Method	NDCG@10	P@10	MRR	R@1000	NDCG@10	P@10	MRR	R@1000
FiQA-2018					DBPedia-Entity			
BM25	23.61	6.34	30.59	73.93	31.80	28.20	58.92	67.60
BM25+Rocchio	19.71	5.86	25.23	75.83	30.76	28.58	57.29	68.19
ChatExp-0	25.04	6.73	32.05	76.19	35.93	30.68	65.57	70.70
ChatExp-Fixed	24.61	6.74	32.04	76.10	36.00	30.70	64.61	71.02
ChatExp-Static	24.67	6.62	32.06	75.99	36.44	30.67	68.19	70.83
ChatExp-Random	25.12	6.77	32.21	76.45	36.64	30.82	66.59	70.68
ChatExp-Cluster	25.11	6.85	32.27	76.38	36.40	30.80	66.83	71.29
ChatExp-NN	25.32	6.94	31.89	76.24	36.33	30.60	66.59	71.16

Table 3: Retrieval performance (%) on FiQA-2018 and DBPedia-Entity. “R@” stands for Recall@.

our automatic variants on three out of four test sets, motivating adaptive example selection.

5.4 Impact of pseudo-labelled pools

All adaptive variants—Static, Random, NN, Cluster—draw demonstrations *exclusively* from the Top-1 pseudo pool. This pool contains no human judgement and only one passage per query, yet provides enough signal:

- On DL’20, **ChatExp-Static** (first four pseudo examples) is already best in recall (78.4) and on par with NN in nDCG, indicating that high-precision top-1 passages are sufficient.
- On FiQA, where the pool is an order of magnitude smaller (5.5 k), **ChatExp-Random** beats Static by sampling varied lexical cues, demonstrating that *quantity can compensate for imperfect relevance*. Random still trails NN/Cluster in ranking metrics.
- On DBPedia, only 64 pseudo examples are available; Cluster and Static thus coincide, and both surpass NN in MRR—implying that

diversity, not pure similarity, matters when the candidate pool is small. Interestingly, both Static and Random outperform NN across all metrics. We hypothesize that with such a limited pool, nearest-neighbor selection suffers from unreliable similarity estimates, highlighting the importance of having a sufficiently large and varied candidate set.

5.5 Nearest-neighbour vs. cluster sampling

The tie between **ChatExp-NN** and **ChatExp-Cluster** follows a clear pattern:

1. NN wins on large pools (DL tracks) because it can always find a topically matched exemplar, reducing semantic drift.
2. Cluster excels on small pools or narrow domains (FiQA MRR, DBPedia Recall@1000) by covering complementary facets and preventing the LLM from over-fitting to one sub-topic.

5.6 Analysis by metric

Recall. All LLM variants increase Recall@1000 over BM25 on every dataset, with a peak gain of

+6.6 points (NN on DL 19). This validates our central hypothesis that query-to-passage generation exposes hidden lexical evidence to the retriever.

MRR and nDCG. Ranking metrics improve less dramatically, sometimes even declining (e.g. NN on FiQA in MRR). Inspection shows that some expansions prepend lengthy background sentences that match semi-relevant documents. A lightweight post-filter or length penalty may mitigate this.

5.7 Comparing Hyper-parameters of ICL

Tables 4 and 5 present a study over two key hyper-parameters in our in-context learning (ICL) prompt: (1) the maximum number of words retained from each demonstration passage, and (2) the number of few-shot examples k . We report Recall@1000 as our primary metric, since the goal is to improve recall for downstream reranking.

Effect of passage length. As shown in Table 4, shorter passages can be surprisingly effective. The **Static** configuration achieves the highest recall (80.07%) at just 40 words, even outperforming longer versions. For **Random**, which introduces lexical variability, longer passages help: recall peaks at 80.55% when using 80 words. Both **Cluster** and **NN** (nearest neighbor) strategies are relatively stable across lengths, showing < 1 point variation. This suggests that when examples are topically aligned, truncation does not substantially hurt, and may help reduce verbosity. Based on these results, we use 60-word passages as default, offering a good trade-off between recall and decoding latency.

Effect of demonstration count. Table 5 explores the number of demonstrations k . We find that larger k does not always improve performance. For example, **Cluster** selection performs best at $k = 2$ (80.61%) but slightly drops at $k = 4$. This may be due to diluted signal when sampling diverse but overly heterogeneous examples. In contrast, **Static** and **Random** benefit from higher k , peaking at 81.11% and 80.74% respectively when $k = 6$. These methods benefit from lexical accumulation across unrelated examples. Interestingly, **NN** performs best at $k = 2$, with diminishing returns as additional examples may introduce non-relevant examples.

We set $k = 4$ and 60-word truncation as default for all main experiments in Section 5. This ensures high recall ($\geq 79.7\%$) with manageable input

Table 4: Recall@1000 across few-shot selection strategies and max passage length.

Strategy	40	60	80
Static	80.07	79.55	79.22
Random	79.88	80.49	80.55
Cluster	79.22	79.25	79.22
NN	79.73	79.00	78.08

Table 5: Recall@1000 across few-shot size k and selection strategy.

Strategy	2	4	6
Static	79.03	79.75	81.11
Random	78.83	80.49	80.74
Cluster	80.61	79.25	80.26
NN	80.04	79.00	77.95

length. These results also confirm that our automatically constructed few-shot pools are robust to such changes and provide consistent gains under different configurations.

6 Conclusion and Future Work

We presented a fully automated pipeline for constructing domain-adaptive few-shot QE candidate datasets and prompting LLMs to perform query-to-passage expansion. Across four benchmarks, the method consistently boosts first-stage retrieval recall, with the four proposed expansion with pseudo examples perform overall the best, showing that without manual effort, in-context learning for QE can still perform well.

For next steps we plan to: (1) integrate learned example retrievers to replace static embedding search; (2) fine-tune lightweight LLM adapters on our pseudo-labelled pairs to reduce inference cost; and (3) couple expansions with re-ranking strategy. We hope these findings encourage broader exploration of in-context learning with LLM for classical IR problems.

Limitations

Our study is confined to passage-level retrieval and a single instruction-tuned model (Qwen-2.5-7B-Instruct). Performance on document-level tasks or with larger LLMs remains to be verified. The pool sizes for DBpedia is relatively small. We also did not explore re-ranking with expanded queries; preliminary experiments suggest that naive con-

catenation can hurt cross-encoder scores, calling for joint training of re-rankers on expanded text. Finally, how to better use the datasets to generate better query expansion can be further investigated.

References

Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard. In *Proceedings of TREC-13*, pages 715–725.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 243–250.

Stéphane Clinchant and Eric Gaussier. 2013. A theoretical analysis of pseudo-relevance feedback models. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, pages 6–13.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. [Overview of the trec 2020 deep learning track](#). *Preprint*, arXiv:2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. Overview of the trec 2019 deep learning track.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, and 1 others. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

Katikapalli Subramanyam Kalyan. 2024. A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, 6:100048.

Yuqi Liu. 2022. Simple yet effective pseudo relevance feedback with rocchio’s technique and text classification. Master’s thesis, University of Waterloo.

Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. 2023. Generative relevance feedback with large language models. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 2026–2031.

Jun Miao, Jimmy Xiangji Huang, and Zheng Ye. 2012. Proximity-based rocchio’s model for pseudo relevance. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 535–544.

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

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 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)*.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. Gpl: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423.

Liang Wang, Nan Yang, and Furu Wei. 2024. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using lucene. *Journal of Data and Information Quality (JDIQ)*, 10(4):1–20.

A Algorithm Details

This appendix provides the pseudocode for our two exemplar selection strategies used in few-shot prompting. These procedures operate over a pool of pseudo-labeled query–passage pairs constructed via our retrieval-based pipeline (see Section 3.2.2). Given a new test query, the selection algorithm determines which few-shot examples to include in the in-context prompt provided to the language model.

We precompute embeddings for the candidate pool by applying a sentence-transformer model to the concatenated string “query + passage” for each example. For a new test query, we encode only the query text.

Algorithm 1 describes the embedding-based nearest neighbor strategy, which selects the top- k examples most similar to the test query in embedding space. Algorithm 2 presents the clustering-based strategy, which partitions the candidate pool into k semantic clusters and selects the medoid (center-nearest) example from each cluster to ensure topical diversity. These methods are compared empirically in Section 5.

Algorithm 1: Embedding-based Example Selection

Input: Query q ; example pool $\mathcal{E} = \{(q_i, p_i)\}$; embedding model $f(\cdot)$; number of shots k

Output: Selected example subset $\mathcal{E}_k$

```

1  $\mathbf{v}_q \leftarrow f(q)$  // Encode the test query only
2 foreach  $(q_i, p_i) \in \mathcal{E}$  do
3    $\mathbf{v}_i \leftarrow f(q_i + p_i)$  // Precomputed embedding for each example
    $s_i \leftarrow \cos(\mathbf{v}_q, \mathbf{v}_i)$ 
4  $\mathcal{E}_k \leftarrow$  top- $k$  examples in  $\mathcal{E}$  ranked by  $s_i$ ;
5 return  $\mathcal{E}_k$ 
```

Algorithm 2: Cluster-based Selection of k Representative Examples

Input: Example pool $\mathcal{E} = \{(q_i, p_i)\}$;

embedding model $f(\cdot)$;

number of clusters k

Output: Cluster-diverse subset $\mathcal{E}_k$

```

1 foreach  $(q_i, p_i) \in \mathcal{E}$  do
2   Encode  $\mathbf{v}_i \leftarrow f(q_i + p_i)$ ; // Joint query-passage embedding
3 Normalize all  $\mathbf{v}_i$  to unit length;
4 Run  $k$ -means clustering on  $\{\mathbf{v}_i\}$ , yielding centroids  $\{\mathbf{c}_1, \dots, \mathbf{c}_k\}$ ;
5 for  $j = 1$  to  $k$  do
6   Let  $C_j$  be the set of examples in the  $j$ -th cluster;
7   Select  $(q^*, p^*) \in C_j$  closest to  $\mathbf{c}_j$  in  $\ell_2$  distance;
8   Add  $(q^*, p^*)$  to  $\mathcal{E}_k$ 
9 return  $\mathcal{E}_k$ 
```

B Examples of the Generated Datasets

Table 6 show examples of the generated query pseudo-relevant passage dataset, which can be used as in-context learning demonstration sampling pool.

Query ID	Query	Pseudo-relevant Passage
879212	<i>what movie did leonardo dicaprio first start out in</i>	List of the best Leonardo DiCaprio produced movies, with trailers of the films when available. All the top movies produced by Leonardo DiCaprio are listed here by popularity, so only highly rated Leonardo DiCaprio films are at the top of the list. If you think the greatest film that Leonardo DiCaprio produced isn't as high as it should be than upvote it so it has the chance to become number one.
928214	<i>what year was sunbelt rentals founded</i>	Sunbelt Rentals supplies the rental equipment and tools for construction projects. Sunbelt Rentals was founded in 1983. Sunbelt Rentals's Headquarters are located at 2341 Deerfield Drive, Fort Mill, South Carolina, USA 29715. Some of Sunbelt Rentals's latest acquisitions include Pride Equipment Corporation, Rental Division, Tower Tech Inc., and Equipment Rental Division, ECM Energy Services, Inc..
241700	<i>how long can chicken live without head</i>	RE: How long can a chicken live without its head? The following is from Livescience.com concerning the myth about chickens living without a head: "True, and not just for a few minutes.A chicken can stagger around without its noggin because the brain stem, often left partially intact after a beheading, controls most of its reflexes.One robust fellow lived... show more The following is from Livescience.com concerning the myth about chickens living without a head: True, and not just for a few minutes. ch...
889816	<i>what qualifications do i need to become a forensic investigator</i>	Q: What qualifications do you need to become a forensic scientist? A: The qualifications you need to become a forensic scientist involve you earning a bachelor's degree in biology, forensic science or chemistry. Some crime scene investigators and forensic science technicians are trained as police officers who have graduated from police academies.
1064278	<i>why do cells respire anaerobically</i>	Best Answer: Anaerobic respiration occurs in muscle cells when they do not have access to enough oxygen to complete aerobic respiration to generate all the ATP they need. In human muscle cells, the end product is lactate/lactic acid. Anaerobic respiration occurs when cells do not have enough Oxygen to undergo the process of aerobic respiration. In animal cells (like humans) anaerobic respiration happens mostly on muscle cells through a process called Fermentation that happens outside the Mitochondria.

Table 6: Randomly sampled examples from the generated dataset. Passages shown here are truncated at 512 characters.