Synthetic Data Generation for Dense Retrieval: A Comparison of LLM Prompting and Agentic Approaches

by

Maxime Bouthillier

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters in Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2026

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Training effective dense retrieval models requires large labeled datasets of query-passage pairs. Creating such pairs is costly and time-consuming to produce, especially in specialized domain settings, as manual annotation is required. Generating synthetic training data using Large Language Models (LLMs) offers a cost- and time-effective alternative. In this paper, we develop and evaluate five synthetic query generation methods using different prompting and agentic frameworks, in conjunction with two hard negative passage generation methods. This is done using the Llama 3.1 8 Billion instruct LLM with the MS MARCO corpus dataset. Bi-encoder dense retriever models are independently trained on each synthetically generated dataset and benchmarked against a baseline model trained on the original in-domain query data from the MS MARCO dataset. The nDCG@10, MRR@10 and Recall@100 are used as evaluation metrics. The best performing synthetic model, trained on Few-Shot generated queries using 8 in-domain examples recovers approximately 89% of the base model's nDCG@10 score. In contrast, agentic workflows do not meaningfully improve over non-agentic generation counterparts. Moreover, the inclusion of synthetically generated hard negative do not increase dense retrieval model learning outcomes. Taken together, these results demonstrate that few-shot LLM prompting is a practical and effective synthetic data generation strategy for dense retriever training in the absence of labeled query data. This can be broadly applied to various domain settings, such as medical notes, industrial proprietary information, or legal documents, where annotated query-passage pairs are scarce.

Acknowledgements

I would like to thank Dr. Giang Tran and Dr. Saeed Ghadimi for all of their guidance and support throughout my masters degree. I have learned so much more than I could have imagined because of them, and I am incredibly grateful to have had this opportunity. I would also like to thank my family, friends, and especially my parents for supporting me throughout this degree.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information retrieval (IR) is a core Natural Language Processing (NLP) task, which consists of finding relevant passages in a corpus of documents given a query. Retrieval tasks can be accomplished using various techniques, such as Boolean retrieval, Query Likelihood models, Term Frequency-Inverse Document Frequency (TF-IDF), and Best Match 25 (BM25) [16][18][21][19]. These retrievers are considered to be sparse, as they only consider word identity and frequency overlaps within a document, and require no training. As a result, sparse retrievers do not capture semantic relationships or context of document passages and queries, limiting their search abilities to symbolic and lexical matches. The advent of transformer-based encoder models, including Bidirectional Encoder Representations from Transformers (BERT) and Universal Sentence Encoder (USE), allowed for dense numerical representation of text [7][5]. Encoder models capture word positioning and semantic relationships using positional word encoding and multi-head attention [7][25]. This returns a high dimensional numerical vector representation of a document, which is referred to as an embedding. Embeddings have allowed for semantic information retrieval. However, unlike sparse retrievers, embedding based dense retrievers require training and fine-tuning using labeled datasets to effectively perform the desired retrieval tasks.

Effective training of dense retrieval models requires a large labeled dataset consisting of queries and passages [22][11]. Labeled retriever datasets consist of positive query-passage pairs annotated with relevance judgments, indicating whether a passage answers a given query. For retriever models to generalize well within and outside their corpus domain, large amounts of query-passage pairs are required to effectively fine-tune the encoder models on the semantic structures of queries and passages [11]. Fine-tuning helps encoder models learn the semantic nuances needed to distinguish relevant passages from irrelevant ones. However, the development of labeled query-passage datasets requires numerous manual

annotators and hours of work [20][22][2]. This problem is exacerbated in domain specific settings, such as medical notes or legal documents, that require qualified manual annotators to develop the dataset [23]. Eliminating the need for manual annotation would lower the barrier to building domain-specific retrieval systems.

The advent of generative Large Language Models (LLMs) has allowed for the creation of human-like text [4]. LLMs are trained on vast corpora of documents, with billions of learned parameters, and can be guided through carefully constructed prompts to produce quality text [4]. Accordingly, LLMs have been utilized for NLP data augmentation and synthetic data creation to enhance training examples [27] [13][9]. Works such as Promptagator and InPars have explored synthetic query generation for information retrieval tasks, and have demonstrated their effectiveness in increasing retrieval performance [3][6]. However, these methods are limited in their exploration of diverse prompting strategies and agentic LLM workflows for synthetic data generation. An agentic LLM workflow is an autonomously executing multi-step data generation pipeline, where LLMs act as agents with specified roles and prompts. Consequently, a thorough exploration of diverse prompting strategies and agentic workflows for synthetic IR data creation remains an open research problem.

In this paper, we explore and evaluate various LLM prompting and Agentic frameworks to generate synthetic data to train bi-encoder dense retriever models. The objective is to compare the effects of different synthetic data generation methods when in-domain query data is missing for dense retriever training. We explore 5 different synthetic query generation methods—Zero-Shot, Few-Shot, Agentic Zero-Shot, Agentic Few-Shot and Agentic Chain-of-thoughts— alongside 2 in-batch hard negative passage generation methods. Retrieval performance of bi-encoders trained on the generated datasets is compared against a baseline trained on the original in-domain MS MARCO dataset. Results demonstrate that Few-Shot prompting, using 8 query-passage pair examples, yields the best synthetically trained model. Furthermore, findings suggest that agentic frameworks do not provide meaningful improvements over their non-agentic counterparts, and that generated hard negative passages severely degrade bi-encoder training outcomes when paired with synthetic query data. The contributions of this paper are as follows:

- Five synthetic query generation pipelines spanning standard prompting and multi-prompt agentic frameworks, generated using the MS MARCO corpus
- Two zero-shot hard negative passage generation methods, generated in conjunction with each synthetic query passage pairs
- Empirical comparison of 18 bi-encoder models, one per dataset configuration, benchmarked against a baseline trained on the original MS MARCO training data

Chapter 2

Related Works

The emergence of transformer-based encoder models has driven a shift from sparse to dense approaches in information retrieval. Karpukhin *et al.* introduced Dense Passage Retrieval (DPR), demonstrating that a dual-encoder (Bi-encoder) architecture trained on query-passage pairs using contrastive learning (via Info NCE loss) can substantially outperform sparse baselines [11]. DPR established the foundational bi-encoder training setup: one encoder encodes queries, and another encodes passages. Batches are created using multiple query-passage pairs, where a passage correctly answers the query, and similarity scores are calculated between all query and passage embeddings. This is optimized over the correct query-passage pairs, with all other entries serving as in-batch negatives—forming the contrastive learning. This research directly builds on this training set up, using the same bi-encoder and contrastive learning approach.

Generating synthetic training data using LLMs for dense retrieval training is an active area of research. Bonifacio *et al.* proposed InPars, a method developing Few-Shot capabilities of large language models to generate synthetic query-document pairs for IR tasks [3]. Cross encoders fine-tuned solely on InPars-generated data were shown to outperform strong sparse baseline methods as well as self-supervised dense retrievers, demonstrating the viability of LLM-based data augmentation. Jeronymo *et al.* subsequently introduced InPars-v2, extending on the works of InPars by replacing proprietary LLMs with open-source alternatives[10]. Furthermore, their work included a reranker model to filter synthetic queries. In-Pars-v2 achieved state-of-the-art results on the BEIR benchmark, highlighting the value of post-generation filtering for synthetic data generation quality.

Promptagator, proposed by Dai *et al.*, frames dense retrieval as a few-shot problem [6]. Given as few as eight task-specific query-passage examples, Promptagator prompts

a FLAN LLM to generate task-specific synthetic queries. A quality generation method is also employed, retaining only queries whose correct positive passage is retrieved by an initial retriever trained on the generated synthetic query data. Promptagator with only 8 annotated examples was shown to outperform significantly larger models trained on MS MARCO across 11 retrieval benchmarks. In 2025, Kim and Baek developed Syntreiver, a two-stage retriever training framework that uses synthetic data generated from black-box LLMs [13]. During the distillation learning stage, Chain-of-Thought prompting is used to synthesize relevant and plausibly irrelevant passages alongside augmented queries to enhance training outcomes. Subsequently, during the alignment stage, the retriever is further fine-tuned using LLM preference signals via Plackett-Luce ranking. Their method surpassed nDCG@K scores across numerous BEIR benchmarks. Kulkarni *et al.* explored LLM-augmented data generation specifically in the dataset search domain. They proposed a pipeline that generates synthetic queries from dataset descriptions to fine-tune dense re-ranking models [15].

A comprehensive survey of the agentic LLM landscape, written by Plaat *et al.*, organizes the growing literature along three aces: reasoning and retrieval, action, and tool use, and multi-agent interaction [17]. Their survey highlights retrieval as a core enabling capability for agentic decision making, directly motivating their investigation of agentic frameworks in synthetic data generation. Matrix, proposed by Wang *et al.* is a decentralized peer-to-peer framework for scalable multi-agent synthetic data generation [26]. Matrix eliminates passing, allowing each agent to progress independently. Their framework achieves substantial quality across diverse data generation tasks, underscoring the capabilities of agentic workflows for synthetic data creation.

Chapter 3

Methodology

3.1 Bi-Encoder Training

Dense retrieval via bi-encoder employs the use of two independent encoder models fine-tuned for passage retrieval. One encoder, E_q , is used to encode queries, while the other E_p, encodes passages. Both encoders, initialized from the `bert-base-uncased` model, are fine-tuned to project their respective inputs such that semantically relevant query-passage pairs are geometrically proximate [7]. The fine-tuning of the encoders is accomplished using known training pairs $\mathcal{I} = \{q_i, p_i\}_{i=1}^{M}$ to which the passage p_i correctly answers the query question q_i. From the M total training pairs, training batches of size B are constructed by respectively encoding each pair item using E_q and E_p. Their embedding outputs are subsequently $L2$ normalized, providing similarity scores bounded between $[-1, 1]$. Specifically, denote

$$E_q(q_i) = \tilde{\mathbf{u}}_i, \qquad E_p(p_i) = \tilde{\mathbf{v}}_i,$$
$$\mathbf{u}_i = \frac{\tilde{\mathbf{u}}_i}{\|\tilde{\mathbf{u}}_i\|_2}, \qquad \text{and } \mathbf{v}_i = \frac{\tilde{\mathbf{v}}_i}{\|\tilde{\mathbf{v}}_i\|_2}$$

A similarity matrix $\mathbf{S} \in \mathbb{R}^{B \times B}$ is then constructed using all query-passage similarities within the batch B,

$$\mathbf{S}_{ij} = \frac{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}{\tau}, \quad \forall i, j = 1, ..., B$$

Positive pairs, representing the correct passage answering the query, naturally lie on the diagonal of $\mathbf{S}$, with all other entries serving as implicit in-batch negatives. The temperature hyperparameter τ is used to help learning outcomes by controlling the concentration of embeddings. The contrastive loss function, InfoNCE, then calculates the diagonal loss using the entries of the similarity matrix $\mathbf{S}$ [24],

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\mathbf{S}_{ii})}{\sum_{j=1}^{B} \exp(\mathbf{S}_{ij})}. \tag{3.1}$$

InfoNCE is derived from Negative Log-likelihood, in which we are given a query $\mathbf{u}$ and a set of passages $\{\mathbf{v}_1, ..., \mathbf{v}_i, ..., \mathbf{v}_B\}$ with $\mathbf{v}_i$ representing a positive passage entry to the query $\mathbf{u}$. We define a categorical distribution using softmax:

$$P(\mathbf{v}_i|\mathbf{u}) = \frac{\exp(s(\mathbf{u}, \mathbf{v}_i)/\tau)}{\sum_{j=1}^{B} \exp(s(\mathbf{u}, \mathbf{v}_j)/\tau}.$$

Maximizing the probability of the correct passage p_i, we minimize the Negative Log likelihood:

$$\mathcal{L} = -\log P(\mathbf{v}_i|\mathbf{u}) = -\log \frac{\exp(s(\mathbf{u}, \mathbf{v}_i)/\tau)}{\sum_{j=1}^{B} \exp(s(\mathbf{u}, \mathbf{v}_j)/\tau}.$$

This is extended to all diagonal entries of the similarity matrix $\mathbf{S}$, and averaged, returning the InfoNCE Loss function (3.1).

The Adaptive Moment Estimation (ADAM) optimizer is used to update the weights of the bi-encoder models during training using the backpropagation gradients of the InfoNCE loss [14]. This optimizer adapts the learning rate per parameter by leveraging estimates of first and second moments of the gradients. Given a gradient at step g_t at step t, we define the momentum as:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2. \end{aligned}$$

These are corrected for bias:

$$\hat{m}_t = \frac{m_t}{1-\beta_1},$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2}.$$

This results in the parameter value update:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v_q}} + \epsilon}.$$

Here $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are specified to the default decay rates, and the stability constant ϵ is set to it's recommended value of 10^{-8} [14].

3.2 Hyperparameter Search

To identify the ideal τ, learning rate and batch size hyperparameters, a preliminary bi-encoder model grid search over training loss was searched. This preliminary search analyzed the training loss convergence on a reduced MS MARCO dataset (containing only 100,000 queries) over 10,000 steps. Five τ temperatures and 5 learning rates are searched, depicted in Figure 3.1. The combination of $\tau = 0.03$ and a learning rate of 2×10^{-5} result in the smallest average loss over the last 5,000 steps, that of 0.0209, with noticeable convergence nearing 0. Note that in this grid search, a default batch size of $B = 256$ was used.

Subsequently, to confirm the choice of batch size a secondary grid search explored $B \in \{32, 64, 128, 256, 512\}$. Figure 3.2 contains the training loss results using the previously defined batch size values. The batch size of $B = 256$ returns the smallest loss over the last 5,000 steps, that of 0.0208. This confirms the ideal batch size to be $B = 256$.

These hyperparameters are used to train all subsequent models as to ensure a fair comparison of training outcomes.

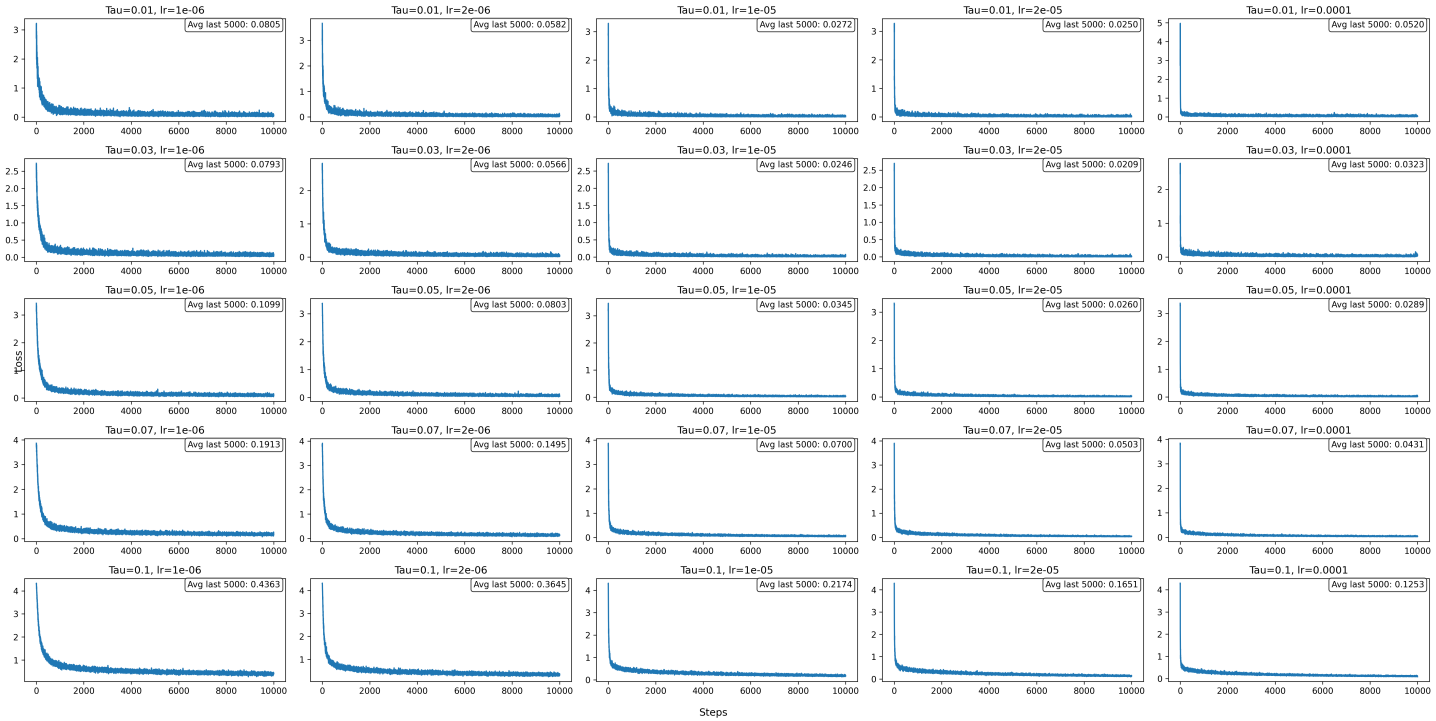

Figure 3.1: Preliminary bi-encoder temperature and Learning Rate Grid Search

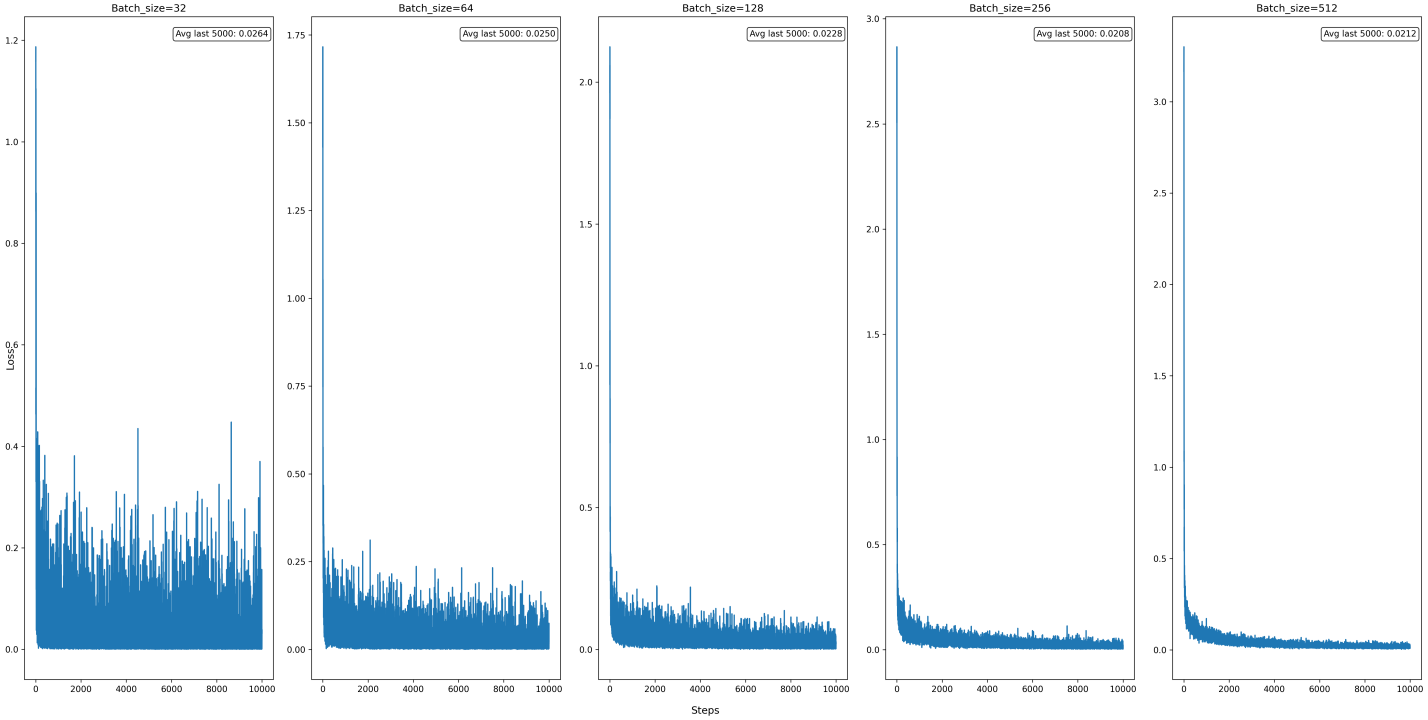

Figure 3.2: Preliminary bi-encoder Batch Size Search

3.3 Bi-Encoder Retrieval

Once a bi-encoder model has been fine-tuned, the entire corpus of documents is encoded using the tuned E_p passage encoder. Formalized, given $\mathcal{P} = \{p_1, ..., p_N\}$, and passage encoder $E_p : \{p_1, ..., p_N\} \rightarrow \mathbb{R}^d$, all passages are encoded and $L2$ normalize to form a matrix $\mathbf{V} \in \mathbb{R}^{N \times d}$,

$$E_p(p_i) = \tilde{\mathbf{v}}_i \quad \forall\ p_i \in \mathcal{P},$$
$$\mathbf{V} = \text{concat}\Big(\frac{\tilde{\mathbf{v}}_1}{\|\tilde{\mathbf{v}}_1\|_2}, ..., \frac{\mathbf{v}_N}{\|\mathbf{v}_N\|_2}\Big).$$

This matrix is then converted into a vector database. This is done using a Facebook AI Similarity Search (`FAISS`) index, which compactly stores the row entries of $\mathbf{V}$, and facilitates fast similarity (inner product) searches with its entries [8]. We specify the `IndexFLATIP` method, which calculates the exact inner product with all entries. This method is slower, but allows for more exact retrieval and does not require additional training. This method also preserves the indexing order, that is, row entry $\mathbf{V}_k$ corresponds to the row entry k within the faiss index's vector database, and furthermore corresponds with passage $p_k \in \mathcal{P}$.

The query encoder, $E_q : \{q\} \rightarrow \mathbb{R}^d$ is thereafter used to encode queries q as they are received. The similarity of the L_2 normalized query embedding $\mathbf{u}$ is calculated with the row matrix $\mathbf{V}$ (faiss index entries) resulting in the similarity vector $\mathbf{s}$,

$$\mathbf{s} = \mathbf{u}\mathbf{V}^T \in \mathbb{R}^N.$$

The top k passages $p_i \in \mathcal{P}$ are subsequently retrieved by selecting the indices i with the highest similarity scores s_i. This comprises the bi-encoder dense retrieval system.

3.4 Synthetic Query Generation

Five synthetic query generation methods are developed. All models are created using the Llama 3.1 8 Billion Instruct LLM [1].

1. Zero-Shot
2. Few-Shot
3. Agentic Zero-Shot
4. Agentic Few-Shot
5. Agentic Chain-of-Thought

For each synthetic query generation method, passages are sampled from the MS MARCO corpus and used to generate synthetic queries. A new dataset is built from the generated queries, alongside the MS MARCO corpus and new qrels—query relations relevance scores, mapping the synthetic queries to their source passages to form training pairs. Each dataset is subsequently used to independently train a bi-encoder, which is then evaluated on the MS MARCO validation set.

3.4.1 Zero-Shot

Let $\mathcal{M}(s_Z, c) \rightarrow \hat{q}$ be the Llama 3.1 8 Billion Instruct model given the system prompt s_Z and context prompt c, returning the synthetic query $\hat{q}$. For Zero-Shot generation, c consists of a passage $p_i \in \mathcal{P}$, and system prompt instructions s_Z. Below is the zero-shot system prompt.

Zero-Shot System Prompt

```
You are a subject matter expert in your field with substantial accumulated
knowledge in a specific subject or topic, validated by academic degrees,
certifications, and/or years of professional experience in that field.

Write a question that elaborates on the provided passage(s).  Ensure that
your question is answered by the passage(s).

Provide only the question and format it as follows:**question**.
```

Outputs are immediately processed to extract the query delimited by double asterisks—producing $\hat{q}$.

3.4.2 Few-Shot

Few-Shot query generation constructs s to contain 8 examples from the MS MARCO dataset. 8 query-passage pairs $\mathcal{F} = \{(q_i, p_i)\}_{i=1}^{8}$, where the passage p_i answers query q_i, are taken from the original dataset, and included within the system prompt s_F to serve as examples. The few-shot prompt follows.

Few-Shot System Prompt

```
You are a subject matter expert in your field with substantial accumulated
knowledge in a specific subject or topic, validated by academic degrees,
certifications, and/or years of professional experience in that field.

Write a question that is concise and directly answered by the provided
passage(s).  Provide only the question and format it as follows:
**question**

Here are some examples:  ℱ
```

Outputs are again immediately processed to extract the query delimited by double asterisks—producing $\hat{q}$. Furthermore, $\mathcal{F}$ is filled with 8 query-passage pairs extracted from the MS MARCO dataset. Note that in instances where multiple passages answer a query, the example pair includes all relevant passages (i.e,. $\mathcal{F}_j = (q_j, p_{j1}, ..., p_{jk})$. Furthermore, at generation time, the passages used to construct the examples $\mathcal{F}$ are withheld so as to not create a synthetic query for an exemplar passage.

3.4.3 Agentic Zero-Shot

The Agentic Zero-Shot query generation consists of two separate LLM calls. The first consists of prompting the LLM as done in Zero-Shot generation (Section 3.4.1), producing the output $\mathcal{M}_1(s_Z, c_1) = \hat{q}$ where $c_1 = p_i$. A second judge LLM $\mathcal{M}_2$ is then given the context prompt $c_2 = (\hat{q}, p_i)$ and system prompt s_J which asks the model to validate whether the synthetic query $\hat{q}$ is answered by the contents of the passage p_i. Below is the judge system prompt used.

Judge System Prompt

```
You are a subject matter expert in your field with substantial accumulated
knowledge in a specific subject or topic, validated by academic degrees,
certifications, and/or years of professional experience in that field.

Your task is to judge whether the provided question is answered by the
provided passage(s).  If there are multiple passages, the question have
an answer contained within each passage.

If the question satisfies the condition, ensure your response contains
"TRUE".  Otherwise, ensure your response contains "FALSE".
```

$\mathcal{M}_2$ returns a binary signal; Outputs are filtered such that if it contains "TRUE", it is assigned 1, and if 'FALSE', 0. A synthetic query $\hat{q}$ is then kept if deemed valid by the judge.

3.4.4 Agentic Few-Shot

The Agentic Few-Shot query generation consists of the same set up as Agentic Zero-Shot, however, with the first LLM call $\mathcal{M}_1$ using the Few-Shot system prompt s_F (Section 3.4.2).

3.4.5 Agentic Chain-of-Thought

Finally, the Agentic Chain-of-Thought, or CoT, consists of 3 LLM calls. The first model $\mathcal{M}_1(s_{CoT1}, c_1) = y$ uses the system prompt s_{CoT1} (Section 3.4.5) instructing it to provide CoT examples of how it would create a synthetic query based on the passage $c_1 = p_i$. Below contains the s_{CoT1} system prompt.

Chain-of-Thought System Prompt 1

```
You are a subject matter expert in your field with substantial accumulated
knowledge in a specific subject or topic, validated by academic degrees,
certifications, and/or years of professional experience in that field.

You will be provided passage(s).  Your task is to provide hypothetical
questions which are answered by the contents of the passage(s).
```

```
Provide your questions:
```

The second model $\mathcal{M}(s_{CoT2}, c_2) = \hat{q}$ creates a synthetic query using the context $c_2 = (y, p_i)$ and system prompt s_{CoT2}. This system prompt asks the model to create a synthetic query for the passage $c_2 = p_i$ and to take inspirations from y. We define the s_{CoT2} system prompt as follows.

Chain-of-Thought System Prompt 2

```
You are a subject matter expert in your field with substantial accumulated
knowledge in a specific subject or topic, validated by academic degrees,
certifications, and/or years of professional experience in that field.

Write a single question that elaborates on the provided passage(s).  This
question must be answered by the passage(s).

Moreover, you will be provided example questions to help guide you.

Provide only the question and format it as follows:**question**.
```

The output $\hat{q}$ (filtered to only include what is contained within the double asterisk) is then validated using a third judge agent $\mathcal{M}_3(s_J, c_3), c_3 = (\hat{q}, p_i)$ (Section 3.4.3).

3.5 Synthetic Negative Passages

Hard negatives passages are documents that are contextually and semantically similar to a query-passage pair, however, does not answer the question asked in the query. The inclusion of hard negatives within a training batch makes the training more challenging, as the bi-encoder models are forced to learn more rigorously [29]. Although in-batch implicit negatives are included within a training batch, these implicit in-batch negatives may be very semantically different, making it easy for the bi-encoder model to differentiate them [28].

The inclusion of in-batch hard negatives alters the batch training set up. For each training pair $\mathcal{I} = \{(q_i, p_i)\}_{i=1}^{M}$, a hard negative passage p_i^- is synthetically created making the training triplet $\mathcal{I} = \{(q_i, p_i, p_i^-)\}_{i=1}^{M}$. Taking B triplets to form a batch, the similarity matrix $\mathbf{S}^{B \times 2B}$ is created. The positive entries no longer lie on the diagonal, but on the $\mathbf{S}_{i,2i-1}$ entries. The InfoNCE loss function labels reflected accordingly.

A LLM $\mathcal{M}(s, c) = p^-$ produces an in-batch hard negative passage using a system prompt s and context prompt $c = (\hat{q}, p)$ containing a positive query-passage pair. Its system prompt s contains instructions on how it should generate the hard negative passage. Two generation methods are explored to create the hard negatives: Zero-Shot Negative Passages v1 and Zero-Shot Negative Passages v2. These differ only in the system prompt s. The former informs the LLM that it is a helpful AI, while the latter instructs it that it is an expert. This allows the comparison of a laymen style passage creation against a detailed expert created one in training outcomes. It can be speculated that the former provides easier training examples, while the latter creates examples that are harder to differentiate from the correct passage entry [29]. The v1 and v2 system prompts are defined as follows:

Zero-Shot Negative Passage System Prompt v1

```
You are a helpful AI assistant.  You are to follow the following
instructions:

You will be given a question and passage(s).  Your task is to write a new
passage that does not answer the question.  However, this new passage
that you will create is to be related to the themes of the question and
passage(s).

Provide only the new passage and format it as follows:  **new passage**
```

Zero-Shot Negative Passage System Prompt v2

```
You are a subject matter expert in your field with substantial accumulated
knowledge in a specific subject or topic, validated by academic degrees,
certifications, and/or years of professional experience in that field.

You will be given a question and passage(s).  Your task is to write a new
passage that does not answer the question.  However, this new passage
that you will create is to be related to the themes of the question and
passage(s).

Provide only the new passage and format it as follows:  **new passage**
```

Chapter 4

Experiments

4.1 Datasets

The MS MARCO (Microsoft Machine Reading Comprehension) dataset is an information retrieval dataset consisting of real user queries sampled from Bing search logs, paired with relevant passages extracted from web documents, accessed via Hugging Face [2]. There are 502,939 queries and 8,841,823 passages. The dataset's queries are reduced to contain only the first 100,000 queries. This is done to reduce training time and computational expenses. Accordingly, all new synthetic datasets create a total of 100,000 queries using the passages within the original MS MARCO corpus. For each query generation method, two additional datasets are created incorporating synthetic hard negatives. One synthetic hard negative is created per query, resulting in 100,000 additional passages. These hard negative passages are only used during training and are excluded from the FAISS index vector database. This results in a total of 18 datasets created, detailed in Table 4.1.

Note: hard negative passages are only used during training, and are not used in the construction of the faiss index vector database. Only original passages of the MS MARCO dataset are contained within the vector database.

Table 4.1: Datasets Configurations

Dataset Name	Total Queries	Total Passages	Hard Negative Passages	Hard Negative Version
MSMARCO_r	100,000	8,841,823	0	NA
MSMARCO_r-	100,000	8,941,823	100,000	v1
MSMARCO_r-v2	100,000	8,941,823	100,000	v2
MARCO_ZS	100,000	8,841,823	0	NA
MARCO_ZS-	100,000	8,941,823	100,000	v1
MARCO_ZS-v2	100,000	8,941,823	100,000	v2
MARCO_FS	100,000	8,841,823	0	NA
MARCO_FS-	100,000	8,941,823	100,000	v1
MARCO_FS-v2	100,000	8,941,823	100,000	v2
MARCO_AZS	100,000	8,841,823	0	NA
MARCO_AZS-	100,000	8,941,823	100,000	v1
MARCO_AZS-v2	100,000	8,941,823	100,000	v2
MARCO_AFS	100,000	8,841,823	0	NA
MARCO_AFS-	100,000	8,941,823	100,000	v1
MARCO_AFS-v2	100,000	8,941,823	100,000	v2
MARCO_ACoT	100,000	8,841,823	0	NA
MARCO_ACoT-	100,000	8,941,823	100,000	v1
MARCO_ACoT-v2	100,000	8,941,823	100,000	v2

4.2 Evaluation

For each of the dataset, a bi-encoder model is trained using its data. To evaluate a bi-encoder model, that a faiss index vector database be constructed and stored using the fine-tuned E_p encoder. This represents the largest computational cost given the 8.8 million passage entries. Evaluation queries are subsequently encoded using the fine-tuned query encoder E_q, and retrieved documents are evaluated using evaluation query relation relevance scores (evaluation qrels). This is accomplished using the MS MARCO dataset's development split, as the official evaluation split is not publicly available. The development split contains the same corpus data, alongside 6980 evaluation queries and qrels.

Normalized Discounted Cumulative Gain at 10 (nDCG@10), Mean Reciprocal Rank at 10 (MRR@10) and Recall at 100 (Recall@100) are used as retrieval evaluation metrics and are defined as follows:

$$\text{nDCG@10}(q) = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1[p^+ \in \mathcal{R}(q, 10)]}{\log_2(\text{rank}(p^+, q) + 1)},$$

$$\text{MRR} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \begin{cases} \frac{1}{\text{rank}(p^+, q)} & \text{if } p^+ \in \mathcal{R}(q, 10), \\ 0, & \text{else} \end{cases}$$

$$\text{Recall@100} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} 1[p^+ \in \mathcal{R}(q, 100)].$$

Here $\mathcal{Q}$ represents the set of test queries, and $\mathcal{R}(q, k)$ represents the set of the k highest scoring retrieved passages using the trained bi-encoder. The $\text{rank}(p^+, q)$ is the rank position of the correct passage $p^+ \in \mathcal{R}(q, k)$ given q.

Defined simply, nDCG@10 measures the ranking quality, rewarding systems that place relevant documents higher within the top 10 results. The normalization scales scores between 0 and 1; within the MS MARCO development split, relevance scores are binary. This reduces nDCG to measuring whether relevant documents are ranked highly. MRR@10 only tracks where the first relevant document appears within the top 10. Finally, Recall@100 alots a score of 1, if a relevant passage is contained within the top 100 retrieved documents, and 0 else. All results are averaged across the total number of development queries.

4.3 Results

Table 4.2 compares the performance of the 18 Bi-encoder models trained on their respective dataset on the evaluation set. All models are trained using the same hyperparameters, specified in Appendix A. We use the Base model scores as a benchmark to assess the quality of learning outcomes of models trained on synthetically generated datasets. The highest scoring synthetic model is the Few-Shot model, scoring 0.18441 in nDCG@10, 0.19866 in MRR@10 and 0.69928 in REcall@100. This represents a reduction of ≈ 0.03 in the former 2 metrics, and ≈ 0.05 in the latter against the Base model. The Few-Shot Agentic counterpart scores slightly lower in all metrics compared to it's non-agentic counter part. The lowest scoring model is the Zero-Shot model, scoring 0.16694, 0.1328 and 0.56166 in terms of nDCG@10, MRR@10 and Recall@100, respectively.

Table 4.2: Model Results

		Evaluation Scores		
Model Name	**Dataset**	**nDCG@10**	**MRR@10**	**Recall@100**
Base	MSMARCO	**0.27694**	**0.22543**	**0.74503**
Base -v1	MSMARCO-	0.27202	0.22106	0.74117
Base -v2	MSMARCO-v2	0.27045	0.22037	0.732
Zero-Shot	MARCO_ZS	0.16694	0.1328	0.56166
Zero-Shot -v1	MARCO_ZS-	0.17565	0.14004	0.56978
Zero-Shot -v2	MARCO_ZS-v2	0.18092	0.14453	0.58689
Few-Shot	MARCO_FS	**0.24655**	**0.19866**	**0.69928**
Few-Shot -v1	MARCO_FS-	0.21371	0.17148	0.64127
Few-Shot -v2	MARCO_FS-v2	0.23385	0.18904	0.6752
Agentic Zero-Shot	MARCO_AZS	0.18441	0.14718	0.60758
Agentic Zero-Shot -v1	MARCO_AZS-	0.16494	0.1316	0.55911
Agentic Zero-Shot -v2	MARCO_AZS-v2	0.17373	0.13834	0.57649
Agentic Few-Shot	MARCO_AFS	0.2344	0.18887	0.67834
Agentic Few-Shot -v1	MARCO_AFS-	0.22111	0.17709	0.66153
Agentic Few-Shot -v2	MARCO_AFS-v2	0.23382	0.1878	0.6614
Agentic CoT	MARCO_ACoT	0.18469	0.14631	0.60307
Agentic CoT -v1	MARCO_ACoT-	0.18629	0.14916	0.60189
Agentic CoT -v2	MARCO_ACoT-v2	0.18	0.143	0.59231

Chapter 5

Conclusion

5.1 Discussion

Analyzing the results in Table 4.2, it is apparent that the best performing models are those using a Few-Shot prompting method. Specifically, the Few-Shot model and the Agentic Few-Shot model score significantly higher in comparison to all other synthetic models with respect to the base model. These models score ≈ 0.03 less in nDCG@10 and MRR@10 and ≈ 0.05 less in Recall@100 against the base. This represents a recovery of the base model learning outcomes of $\approx 89\%$ in the former metrics, and $\approx 93\%$ in the latter. With the exception of the Agentic Few-Shot model, which scores comparably, all other synthetic models scored significantly lower in all evaluation metric scores against the base and Few-Shot models. This would suggest that developing 8 in-domain query-passage examples can improve the quality of generated synthetic data and thereby increase the training and performance of bi-encoders. Furthermore, this is supported by Dai *et al.*, who reported that as few as 8 domain examples during synthetic data generation were sufficient to achieve competitive retriever performance [6].

What significantly stands out is the performance of synthetic query models trained with generated in-batch hard negatives. With the exception of the Zero-Shot family, and every so slightly the agentic CoT -v1, all other model families trained with hard negatives score slightly lower in all metrics in comparison to their counterparts trained without them. Moreover, it is worth noting that the Few-Shot negative v1 version scores ≈ 0.03 points lower than the Few-Shot counterpart. There is also minimal difference, generally $\approx 0.01 - 0.02$, between the negative passage creations v1 and v2. This would suggest that the different prompt designs have minimal impact on the quality of the synthetic negatives

created. These findings stand in contrast to the well-established role of hard negatives in dense retrieval training. This would suggest that hard negatives that are generated synthetically in a zero-shot fashion are not grounded in the difficulty distribution of true negative passages within the MS MARCO evaluation dataset. As such, they introduce ambiguous contrastive signals that add noise during training, reducing learning outcomes as opposed to increasing them. The negative generated passages, prompted to be thematically related but non-answering, may too closely mirror the semantic and lexical properties of the positive passage, making the contrastive objective ill-posed for the bi-encoder. This is consistent with observations by Xiong *et al.* who noted that the selection of hard negatives is crucial, as ill selected hard negatives specifically introduce noise within the similarity matrix $\mathbf{S}$ [28]. The Zero-Shot exception may reflect the comparatively lower semantic richness of the zero-shot generated queries data. This would constrain the Llama LLM's hard negative generation, inadvertently creating better suited hard negatives leading to positive learning outcomes, albeit marginal. This contrasts with findings from works including InPars, InPars-v2 and Syntreiver, who reported significant improvement in dense retriever learning outcomes when including hard negative passages [3][10][13]. This would suggest that the benefits of including hard negatives are contingent on their quality and semantic relationships with the positive query passage pair — conditions difficult to guarantee under zero-shot generation, and without explicit quality validation of the negatives.

Lastly, it is apparent that when comparing non-agentic generation methods against agentic frameworks, there are small variations in learning outcomes. To begin, comparing the Zero-Shot and Agentic Zero-Shot, the agentic method score ≈ 0.18441 while the non-agentic method scores 0.16694 in nDCG@10. This is a modest improvement in learning quality outcomes, despite the agentic model presumably creating more reliable data as it is validated by another independent LLM. Comparing agentic and the non-agentic performance of the Few-Shot models, we report a minimal degradation in scores. These results consistently suggest that validation agents do not meaningfully improve bi-encoder training outcomes across prompting strategies despite presumably creating more reliable data across any prompting strategy. Therefore, the additional computational cost of a validation agent is not warranted [17]. Furthermore, it appears that the Agentic Chain-of-Thought method does not increase the synthetic query quality over zero-shot prompting methods. This would imply that the Chain-of-Thought examples provided to the creation prompt (second LLM call in the CoT Agentic scheme), guides the LLM generation in the same direction as in zero-shot prompting. It could be argued that the generation of a synthetic query conditioned on the passage is well conditioned in a zero-shot setting, and that inclusion of CoT examples does not meaningfully impact the synthetic query generation distribution. Once more, this indicates unnecessary computational expenses associated

with an agentic CoT design, as no meaningful benefits are observed despite requiring an additional decoding sequence for each synthetic query.

5.2 Future Research Directions

Building on these findings, several directions remain to be explored. First and foremost, further investigations should examine how examples within a Few-Shot system prompt are to be selected and developed to maximize the generation of synthetic data. This research demonstrated that selecting 8 query-passage pairs significantly increases synthetic data training outcomes; however, these pairs were randomly selected. As such, future research should explore various example collection strategies and their impacts in the quality of generated synthetic data for training.

Secondly, further detailed investigations should examine the generation of Few-Shot synthetic hard negatives. Given the improvements in training outcomes for Few-Shot query generation, Few-Shot negative passages could provide better guidance to LLMs for generation, resulting in better training examples representative of the true negative distribution. However, as it stands, the inclusion of zero-shot hard negatives with synthetic queries leads to conflicting performance outcomes. The relationship between synthetically generated queries while developing synthetically generated hard negatives appears to play a role in the quality of hard negative passages. This is observed with the Zero-Shot model in comparison to all other model families. The Zero-Shot family's inclusion of synthetic hard negatives during training increased learning outcomes, while they decreased all others. Future investigations should study the behavior of this relationship, alongside the exploration of mining BM25 hard negatives to serve as few-shot examples. Additionally, embedding similarity thresholds and keyword matches could provide guardrails to generate stronger hard negatives, steering models to better learning outcomes.

5.3 Conclusion

Training a reliable dense retriever in specialized domain settings has numerous applications. While a corpus of documents is available, the absence of queries and query-relevance mappings (qrels) makes training bi-encoders impossible. Generating synthetic queries using large language models (LLMs) offers a cost- and time-effective solution to this problem. This paper explores the effectiveness of synthetically generated queries for bi-encoder dense retriever training. Synthetically trained models are benchmarked against a base model trained on the true in-domain query data. Keeping all else constant, this setup enables direct comparison of the generated data in terms of both model learning outcomes and retrieval performance in the target domain, evaluated on an in-domain query evaluation set. When comparing methods, results indicate that Few-Shot prompting using 8 query-passage pair examples from the MS MARCO dataset substantially improves synthetic query quality, with the Few-Shot models recovering approximately 89% of the base model's nDCG@10 score. In contrast, Zero-Shot, Agentic Zero-Shot, and Agentic CoT methods lag considerably behind, suggesting that even a small sample of in-domain examples provides crucial guidance for synthetic query generation. Additionally, although judge validation agents are intuitive as a mechanism to reduce hallucinations and ensure query quality, their inclusion did not meaningfully impact learning outcomes despite presumably producing more reliable data. A particularly striking finding is the minimal learning variations of models trained with synthetically generated hard negative passages. This suggests that zero-shot hard negatives introduce an ambiguous and noisy contrastive signal that does not significantly improve bi-encoder training. Although motivated by the challenge of training dense retrievers without labeled data, this work demonstrates that few-shot synthetic query generation is a practical and broadly applicable strategy relevant to any domain where annotated query data is scarce. Hence, few-shot LLM prompting can serve as an effective data generation step for dense retrieval systems requiring domain-specific training pairs.

References

[1] Meta AI. Llama-3.1-8b-instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct, 2024. Accessed: 2026-04-08.

[2] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated MAchine reading COmprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[3] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. InPars: Unsupervised dataset generation for information retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2387–2392. Association for Computing Machinery, 2022.

[4] 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, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

[5] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174. Association for Computational Linguistics, 2018.

[6] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. In *Proceedings of the Eleventh International Conference on Learning Representations*, 2023.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics, 2019.

[8] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

[9] Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for NLP. *arXiv preprint arXiv:2105.03075*, 2021.

[10] Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. InPars-v2: Large language models as efficient dataset generators for information retrieval, 2023.

[11] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics, 2020.

[12] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[13] Minsang Kim and Seungjun Baek. Syntriever: How to train your retriever with synthetic data from llms. In *Findings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Findings of NAACL. Association for Computational Linguistics, April 2025.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[15] Ria Kulkarni et al. Improving dense retrieval models with LLM augmented data for dataset search. *Knowledge-Based Systems*, 2024.

[16] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England, 2008.

[17] Aske Plaat et al. Agentic large language models, a survey, 2025.

[18] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. ACM, 1998.

[19] Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241. Springer-Verlag, 1994.

[20] Paul Röttger, Bertie Vidgen, Dirk Hovy, and Janet Pierrehumbert. Two contrasting data annotation paradigms for subjective NLP tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.

[21] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[22] 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)*, 2021.

[23] UKP Lab. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. https://github.com/beir-cellar/beir, 2021. GitHub repository.

[24] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[26] Dong Wang, Yang Li, Ansong Ni, Ching-Feng Yeh, Youssef Emad, Xinjie Lei, Liam Robbins, Karthik Padthe, Hu Xu, Xian Li, Asli Celikyilmaz, Ramya Raghavendra, Lifei Huang, Carole-Jean Wu, and Shang-Wen Li. Matrix: Peer-to-peer multi-agent synthetic data generation framework, 2025.

[27] Chenxi Whitehouse, Monojit Choudhury, and Alham Fikri Aji. LLM-powered data augmentation for enhanced cross-lingual performance. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 671–686. Association for Computational Linguistics, 2023.

[28] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jiawei Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive estimation for dense text retrieval. In *International Conference on Learning Representations (ICLR)*, 2021.

[29] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, 2021.

APPENDICES

Appendix A

Hyperparameter Configuration

Table A.1 contains the hyperparameter configuration used to train all 18 bi-encoder models.

Table A.1: Hyperparameter Configuration

Hyperparameter	Value
Queries	100,000
τ	0.03
Batch-size	256
Steps	400
Optimizer	ADAM
Learning Rate	2e-5
β_1	0.9
β_2	0.999
ϵ	10^{-8}