Retrieval-Oriented Pre-training Outperforms Language Specialization: Evidence from Bengali University FAQ Retrieval

by

Mithila Arman
23266024

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
May 2026

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Mithila Arman

23266024

Approval

The thesis titled "Retrieval-Oriented Pre-training Outperforms Language Specialization: Evidence from Bengali University FAQ Retrieval" submitted by

1. Mithila Arman (23266024)

Of Spring, 2026 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science and Engineering on May 16, 2026.

Examining Committee:

Supervisor:
(Member)

Dr. Farig Yousuf Sadeque

Associate Professor
Department of Computer Science and Engineering
BRAC University

Internal Examiner:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
BRAC University

External Examiner:
(Member)

Mr. Md. Moinul Hoque

Associate Professor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

M.Sc. Thesis Coordinator

Dr. Md Sadek Ferdous

Professor
Department of Computer Science and Engineering
BRAC University

Ethics Statement

Title: Retrieval-Oriented Pre-training Outperforms Language Specialization: Evidence from Bengali University FAQ Retrieval
Author: Mithila Arman
Supervisor: Dr. Farig Yousuf Sadeque
Department: Department of Computer Science and Engineering
University: BRAC University

1. Ethical Approval:
This thesis received approval from the aforementioned examining committee to ensure adherence to ethical guidelined.

2. Data Management:
All data used in this thesis created manually in compliance with data protection regulations, stored securely, and retained according to institutional policies.

Abstract

Approaching Bengali institutional FAQ answering as a dense bi-encoder retrieval task, this study introduces University FAQs, a real-world benchmark. The dataset, extracted from 10,264 raw records, is cleaned up to obtain data on 7,233 question–answer pairs distributed among 148 topic labels that allow us the systematic evaluation of a total of 18 sentence encoders including Bengali specific sentence encoders, multilingual SBERT variants and E5 retrievers the BGE-M3 as well as GTE considered in this work along with several MLM based baselines under a unified fine-tuning framework with Multiple Negatives Ranking Loss. This study moves beyond traditional retrieval metrics by also evaluating ranking behaviour, semantic-space quality, calibration, efficiency and levels of resistance to character-level noise with a view to defining explicit realistic deployment requirements of educational support systems. On a broad level, the results demonstrate retrieval-oriented multilingual contrastive encoders outperform Bengali specific and MLM-only baselines signal, clearly and consistently. Out of all models, multilingual-e5-large performs the best overall (Acc@1 = 0.923, MRR@10 = 0.949), with BGE-M3 and multilingual-e5-large-instruct in close second and third place respectively. This study analysis also demonstrates that contrastive pre-training leads to a mean Acc@1 gain of 0.175 over MLM-only models, whilst multilingual-e5-small provides the best quality–efficiency trade-off in latency-sensitive settings. Moreover, a confidence-based abstention strategy yields 96.4% accuracy while only answering 68.2% of queries. More generally, this work lays a solid empirical foundation for Bengali FAQ retrieval and shows that multilingual retrieval-oriented pre-training may be more beneficial than monolingual specialization for actual low-resource educational question answering.

Keywords: Bengali FAQ Retrieval; Dense Bi-Encoder Retrieval; Multilingual Sentence Encoders; Contrastive Learning; Educational Question Answering; Low-Resource NLP; Bengali Language Technology

Dedication

This thesis is dedicated to my supervisor for his support, encouragement and advice during this research journey. With his support, patience and expertise inspired me to work with dedication and confidence.

Acknowledgement

All praise to the Great Allah for whom my thesis have been completed without any major interruption.
To my thesis supervisor Dr. Farig Yousuf Sadeque sir for his kind support and advice in my work. He helped me whenever I needed help.
The whole examining committee of the thesis.
I would also like to express my heartfelt gratitude to Dr. Haewon Byeon sir, Dr. Samuel Kim sir, and Dr. Huda Fish sir for their guidance, encouragement, and motivation.
And finally to my parents without their throughout sup-port it may not be possible. With their kind support and prayer I am now on the verge of my graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$AdamW$ Adam with Decoupled Weight Decay

AMP Automatic Mixed Precision

$BEIR$ Benchmarking Information Retrieval

$BERT$ Bidirectional Encoder Representations from Transformers

$BFAQ$ Bengali FAQ / BRAC University FAQ Dataset

$BGE\text{-}M3$ BAAI General Embedding M3

$BLUB$ Bangla Language Understanding Benchmark

$BM25$ Best Matching 25

$chrF$ Character n-gram F-score

$CoSENT$ Cosine Sentence Loss

CPU Central Processing Unit

$CUDA$ Compute Unified Device Architecture

$E5$ Multilingual E5 Embedding Model Family

ECE Expected Calibration Error

$ELECTRA$ Efficiently Learning an Encoder that Classifies Token Replacements Accurately

EM Exact Match

$F1$ F1 Score

FAQ Frequently Asked Questions

$FP16$ 16-bit Floating Point

$FP32$ 32-bit Floating Point

GPU Graphics Processing Unit

GTE General Text Embeddings

$InfoNCE$ Information Noise-Contrastive Estimation

$INSTRUCTOR$ Instruction-based Sentence Embedding Model

IR Information Retrieval

KD Knowledge Distillation

$LaBSE$ Language-agnostic BERT Sentence Embedding

LLM Large Language Model

$MIRACL$ Multilingual Information Retrieval Across a Continuum of Languages

MLM Masked Language Modeling

$MMTEB$ Massive Multilingual Text Embedding Benchmark

$MNRL$ Multiple Negatives Ranking Loss

$MPNet$ Masked and Permuted Pre-training Network

$MTEB$ Massive Text Embedding Benchmark

$NLLB$ No Language Left Behind

NLP Natural Language Processing

NLU Natural Language Understanding

PCA Principal Component Analysis

QA Question Answering

$RoBERTa$ Robustly Optimized BERT Pretraining Approach

$SBERT$ Sentence-BERT

$SOTA$ State of the Art

STS Semantic Textual Similarity

$t\text{-}SNE$ t-distributed Stochastic Neighbor Embedding

$T4$ NVIDIA Tesla T4 GPU

TLM Translation Language Modeling

USE Universal Sentence Encoder

$VRAM$ Video Random Access Memory

$XLM\text{-}R$ Cross-lingual Language Model RoBERTa

Chapter 1

Introduction

1.1 Research Design

The increasing use of digital student-support platforms has created a high demand for effective scalable question answering systems in higher education. Many universities receive a huge number of repetitive queries regarding admission, tuition and fees, scholarship, academic regulations, residence hall, dorms and exam and administrative processes on a daily basis. In reality, these queries are often already addressed by institutional FAQ repositories; however, finding the most relevant response still proves to be hard since semantically similar questions can be articulated in a myriad ways. Such challenges are felt more acutely for Bengali since, in comparison to other languages with rich NLP resources, user queries can often have transliterated English words or acronyms and institutional text could be a heavy mixture of domain specific lexicon with real-life natural language expressions of Bengali speakers. Because of these constraints, Bengali FAQ retrieval challenges are both practically important and methodologically non-trivial.[15] [14] [19].

However, the vast majority of past works on Bengali question answering have focused very little on the domain-specific FAQ retrieval task and mostly tackled reading comprehension or answer generation or general language understanding tasks. While current Bangla benchmarks have broadened the resource landscape considerably, they are often designed as extractive or long-context QA tasks where a system chooses an answer span from some passage or reasons over long context. On the other hand, institutional FAQ answering is more accurately captured as a retrieval task over short question–answer pairs, which identifies the archived FAQ entry with intent that closely resembles that of the user's query and returns its corresponding answer. This is a key difference, because the answers to FAQs tend to be non-extractive spans bound to content and more operational in nature. This leads to a gap between the needs of realistic Bengali FAQ retrieval [15] [43] [50] [44] and approaches optimized for passage-level reading comprehension.

We close this gap by recasting Bengali university FAQs answering as a dense bi-encoder retrieval problem over a purpose-built BRAC University FAQ corpus. The dataset consists of 10,264 raw records spread over topic question and answer fields after cleaning deduplication and normalizing data to usable FAQ pairs there are 7,233 usable records retaining 148 topic labels.

In order to systematically examine this task the paper evaluates 18 sentence encoders, including two Bengali-specialized models, a number of multilingual SBERT

baselines, E5 contrastive retrievers and three up-to-date multilingual retrieval generators like BGE-M3 and GTE along with unadorned masked-language-model (MLM) based on top of different pooling heads across languages. This comparison is structured to assess a number of underlying axes at the same time: language-specific versus multilingual pre-training, contrastive versus MLM-pre-training objectives, compact versus large parameter metrics and generic versus instruction-tuned retrieval models. Multiple Negatives Ranking Loss is used for fine-tuning all encoders under a single training protocol which allows us to conduct controlled experiments on the impact of pre-training design and model family on downstream Bengali FAQ retrieval quality [37] [34] [40] [31].

The evaluation protocol utilized by design is also broader than the standard top-k retrieval reporting. In addition to measuring Accuracy@k and MRR@10, the experiments measure ranking-distribution behavior; embedding-space separation, topic clustering tendencies, textual similarity to the gold answer; calibration quality and efficiency; and robustness against character-level noise. By deployment requirements: a production faq system that retrieves the correct answer needs to rank well in the top few, be resilient to misspellings of the query, and more; it needs confidence estimates allow abstaining while navigating the heavy trade-off between accuracy and latency. So the resulting analysis is more than a simple comparison against a leaderboard on what we can cite as a more realistic carrier for what makes an Bengali FAQ retriever useless in practice. [21] [25] [37].

Here are just a few of the key conclusions to be drawn from the findings. First, modern contrastively trained multilingual retrievers are significantly better than MLM-only baselines and even surpass nearly all Bengali-specific models in aggregate on this task. The top model overall, multilingual-e5-large goes to a peak of Acc@1 = 0.923 and MRR@10 = 0.949, followed closely by BGE-M3 and multilingual-e5-large-instruct. Second, retrieval-oriented pre-training is better for any natural language processing (NLP) task than language specificity alone: vastly contrastively trained models can hit an overall mean Acc@1 of 0.853 versus only 0.678 on MLM-only models (effect size +0.175). Third, Bengali specialization is certainly not irrelevant, the Bangla sentence-transformer still competes and takes a fourth place overall but its good performance seems to derive more from sentence-level similarity training than monolingual MLM pre-training on its own. Ultimately, the study outlines clean deployment trade-offs: multilingual-e5-small provides the highest quality-per-parameter ratio from amongst state-of-the-art models and a confidence-based abstention rule can achieve 96.4% precision while answering 68.2% of queries [37] [34].

This work supplies a realistic Bengali educational FAQ benchmark and systematic empirical analysis of dense retrieval design choices for low-resource institutional QA. It more generally challenges the prevailing belief that monolingual datasets should provide a relative advantage to monolingual pre-training, and it shows more specifically that generic large-scale multilingual contrastive retrieval models can deliver improved re-ranking performance on question-and-answer pairs for Bengali FAQ retrieval if they are evaluated as well as fine-tuned with a suitable retrieval perspective.

1.2 Scope and Significance of the Study

This work makes the following contributions.

- Unlike extractive or generative QA methods, we treat FAQ answering as a dense bi-encoder retrieval task which matches well with how institutional FAQ repositories are setup: answers are normally short, self-contained and operationally useful.

- This paper introduces and analyzes a multi-faceted, realistic Bengali educational FAQ benchmark (the BFAQ), consisting of 10,264 raw records and 7,233 cleaned records categorized in to 148 topic labels extracted from BRAC University. We also characterise the corpus based on completeness, topic skew, lexical overlap, word and character statistics, script composition and tokenized-length budget providing a solid empirical baselines for Bengali FAQ and educational NLP.

- We evaluate 18 sentence encoders specifically designed for Bengali, as well as multilingual SBERT models, E5 retrievers, purely modern multilingual retrievers design and generic multilingual MLM baselines in large scale controlled experiments. This study is performed in the most systematic way as far as framing of the manuscript could do.

- We leverage Multiple Negatives Ranking Loss through unification of training under the same fine-tuning framework, allowing pre-training objectives and architectures to be effectively compared against each other when trained under identical downstream supervision. We also provide ablations on individual design choices including zero-shot vs. fine-tuned performance, sequence length, batch size and pooling strategy.

- We take a comprehensive evaluation protocol that goes beyond common retrieval metrics with respect to ranking-distribution statistics, quality-of-semantic-space, calibration, character-noise robustness, text similarity with gold answers and efficiency. This gives a deployment-oriented point of view on Bengali FAQ retrieval and identifies not only the best model, but also why it is good and in what prerequisites.

- We demonstrate that state-of-the-art contrastively trained multilingual retrievers outperform virtually all Bengali-specific and MLM-only baselines on this task. Notably, multilingual-e5-large obtains the highest overall performance with Acc@1 = 0.923 and MRR@10 = 0.949; contrastive pre-training gives a +0.175 gain in mean Acc@1 over MLM-only models (with differences again obtained by paired Student t-tests). These results exemplify that retrieval-agnostic objective design could be more important than strict monolingual pre-training for Bengali FAQ search.

- Identify pragmatic deployment suggestions for authentic Bengali FAQ systems. Our main results are the following: multilingual-e5-large is our overall best model, multilingual-e5-small provides the optimal accuracy-–efficiency trade-off for latency-constrained settings, multilingual-e5-large-instruct has the best calibration properties, and an abstention threshold of $\tau_{\text{abs}} = 0.72$ provides a precision of 96.4% on answered queries.

Chapter 2

Literature Review

The present study is positioned between Bengali question answering, multilingual sentence representation learning and domain-specific FAQ retrieval. We detail our work on Bengali university FAQ answering as a dense bi-encoder retrieval problem over an annotated BRAC University FAQ corpus, accompanied by systematic comparison across 18 Bengali-specific and multilingual encoders in the uploaded manuscript.

Recent works in Bengali QA mainly focused on expanding the resource space, rather than directly contributing to FAQ retrieval. One of the first large Bangla reading-compehension benchmark along with various question and answer types was created by BanglaRQA (Ekram et al., 2022) [15] while a robust Bangla-centered encoder pre-trained on a different context as well as a formal benchmark suite expanding Tangential Natural Language Understanding (NLU) evaluation were provided in BanglaBERT (Bhattacharjee et al., 2022) [14]. This resource-oriented path proceeds with BanglaNLG and BanglaT5 (Bhattacharjee et al., 2023) [18] enhancing bangla generation benchmarks and Question Answer Generation in Bengali (Shahriar et al., 2023) [22] tackling data-scarcity via synthetic QA generation for bangali. On the other hand, bnContextQA (Ahmad et al., 2025) [43] and NCTB-QA (Eyasir et al., 2026) [50] are long-context Bangla QA based datasets that concentrated on positional retrieval effects and scale-education in Bengali question answering dataset with diversity of un-answerable and adversarial content. As a whole, these studies widen the Bangla QA landscape but mainly focus on reading comprehension, generation or longer-context reasoning level tasks, and hence are neither semantic retrieval over short institutional FAQ pairs.

The other division of the work relates to representation learning for Bangla and Indic. On the other hand, BanglaBERT (Bhattacharjee et al., 2022) [15] focused on monolingual Bangla pretraining, whereas Associative in itself Indic paradiastically and IndicBERT v2 (Doddapaneni et al., 2023) [19] take to wider view of a multilingual Indic structure and testify that Rampayat phenomenon after transaltion wasts achieves superior task super crosswise parlence. BanglaEmbed (Kabir et al., 2024) [32] is especially relevant at the sentence level since we are directly applying multilingual cross-lingual distillation to provide efficient Bangla sentence embeddings, which gets us closer to the representation regime needed for semantic retrieval. At instruction-tuning level, COMMIT (Lee et al., 2024) [33] demonstrated that code-mixed multilingual instruction tuning can substantially improve low-resource QA, while ContrastiveMix (Do et al., 2024) [29] showed that code-mixing has non-trivial

impact on retrieval quality in retrieving information across multiple languages and should be dealt cautiously in multilingual IR. These findings directly pertain to the corpus of the uploaded manuscript which contains an understandingable yet non-negligible amount of Latin-script and transliterated content.
Outside of Bangla, the retrieval and embeddings evaluation ecosystem has also evolved fast. Related Work MTEB (Muennighoff et al., 2023) [21] created a wide-ranging multi-task benchmark of embedding tasks and included 112 languages, whereas MIRACL (Zhang et al., 2023) [25] provided a multilingual ad hoc retrieval benchmark with human relevance judgements by native speakers in 18 different languages. The difficulty of zero-shot retrieval in a large South Asian language was shown through Hindi-BEIR (Acharya et al., 2024) [26] and this was extended to cross-lingual retrieval for Sanskrit by Anveshana (Jagadeeshan et al., 2025) [46]. Even broader was the extension of MMTEB (Enevoldsen et al., 2025) [45], which extended multilingual embedding evaluation (to over 500 tasks, and over 250 cross-lingual languages). All in all, these benchmarks also make it much harder to credibly evaluate multilingual retrieval models on Eenglish-only or sentence-similarity-only testbeds; they should be widely evaluated on heterogeneous languages, domains and retrieval settings.

Table 2.1: Recent related work with reported results and relevance to the present Bengali FAQ retrieval study.

Study	**Reported result(s)**	**Relevance to the present study**
BanglaRQA (Ekram *et al.*, 2022) [15]	BanglaT5: EM 62.42, F1 78.11; BanglaBERT: EM 47.55, F1 63.15	A foundational Bangla QA benchmark with strong Bangla baselines, but it evaluates extractive reading comprehension rather than FAQ-style dense retrieval over archived institutional questions.
BanglaBERT (Bhattacharjee *et al.*, 2022) [14]	BanglaBERT outperforms mBERT and XLM-R by +6.8 and +4.3 BLUB points; BanglishBERT exceeds them by +15.8 and +10.8 in zero-shot transfer	A key Bangla-specific pretrained encoder family for testing whether monolingual Bengali models can compete with multilingual retrievers in FAQ retrieval.

Continued on next page

Table 2.1 continued from previous page

Study	Reported result(s)	Relevance to the present study
BanglaQA / Question Answer Generation in Bengali (Shahriar *et al.*, 2023) [22]	Best model F1 86.14; human upper-bound F1 95.72; NLLB improves BLEU by 44% over prior SOTA	Shows the value of synthetic Bangla QA data generation, but the focus is resource creation rather than semantic retrieval over FAQ pairs.
INSTRUCTOR: One Embedder, Any Task (Su *et al.*, 2023) [23]	Average score 58.4 (335M) and 58.8 (XL) across 70 datasets; 3.4% average improvement over previous best	Established instruction-aware embeddings, which are highly relevant when comparing standard sentence encoders with instruction-tuned retrievers.
GTE: Towards General Text Embeddings with Multi-stage Contrastive Learning (Li *et al.*, 2023) [20]	Full two-stage GTE-large reaches 62.4 on MTEB; pretraining-only 59.0; surpasses INSTRUCTOR-large by 1.5 points	Strong evidence that multi-stage contrastive learning materially improves embedding quality, directly relevant to the MLM-vs-contrastive question in Bengali FAQ retrieval.
Improving Text Embeddings with Large Language Models (Wang *et al.*, 2024) [37]	E5-Mistral-7B achieves 56.9 on BEIR and 66.6 on MTEB; improves previous MTEB SOTA by 2.4 points	Shows how LLM-initialized embedding models can outperform earlier encoder families, helping contextualize the latest retrieval-oriented embedding trend.
Multilingual E5 (Wang *et al.*, 2024) [38]	MTEB scores: multilingual-e5-small 57.9, base 59.5, large 61.5, large-instruct 64.4	One of the most directly relevant multilingual retriever families for the present study, especially because the uploaded paper benchmarks E5 variants on Bengali FAQ retrieval.

Continued on next page

Table 2.1 continued from previous page

Study	Reported result(s)	Relevance to the present study
BGE-M3 (Chen *et al.*, 2024) [28]	MIRACL nDCG@10: Dense 69.2, Sparse 53.9, Multi-vector 70.5	A strong multilingual retriever that unifies dense, sparse, and multi-vector search, making it highly relevant for modern FAQ retrieval comparisons.
Jina-ColBERT-v2 (Jha *et al.*, 2024) [31]	BEIR avg nDCG@10 53.1; MIRACL avg nDCG@10 62.3; mMARCO avg MRR@10 31.3	Demonstrates that carefully engineered multilingual late-interaction retrieval remains highly competitive and is a useful architectural point of comparison.
mGTE (Zhang *et al.*, 2024) [40]	MTEB scores: English 61.40, Chinese 62.72, French 59.79, Polish 58.22	Relevant for understanding long-context multilingual embedding and reranking performance beyond short-text sentence encoders.
SWIM-IR / SWIM-X (Thakur *et al.*, 2024) [35]	SWIM-X(250K) reaches Recall@5kt 60.5 on XOR-Retrieve; SWIM-X(120K)MT reaches avg MRR@10 29.2 on XTREME-UP	Shows that synthetic multilingual retrieval data can substantially improve retrieval quality in low-resource or cross-lingual settings.
COMMIT (Lee *et al.*, 2024) [33]	Improves exact match in low-resource QA by up to 32× on MLQA and XQuAD	Especially relevant for Bengali deployment because real user queries often include code-mixed or transliterated tokens.
bnContextQA (Ahmad *et al.*, 2025) [43]	Gemini-2.5-Flash-Lite achieves best F1 74.17; GPT-4.1-nano reports EM 59.37 and F1 67.07	A recent Bangla QA benchmark for long-context reasoning, useful for situating Bengali QA progress even though the task differs from FAQ retrieval.

Continued on next page

Table 2.1 continued from previous page

Study	Reported result(s)	Relevance to the present study
Hindi-BEIR and NLLB-E5 (Acharya *et al.*, 2025) [42]	NLLB1.3B-E5 obtains average nDCG@10 of 48.48 and reports a 37% average gain over BM25	A highly relevant Indic-language retrieval benchmark that supports the need for South Asian language-specific retrieval evaluation beyond English.
Gemini Embedding (Lee *et al.*, 2025) [47]	MMTEB mean 68.32, outperforming multilingual-e5-large-instruct by +5.09	A very recent strong general embedding model, useful as an upper-end reference point for modern multilingual embedding quality.
NCTB-QA (Eyasir *et al.*, 2026) [50]	BERT improves F1 from 0.150 to 0.620, a 313% relative gain	Educational-domain Bangla QA is thematically close to the present educational FAQ setting, although the task is still textbook QA rather than FAQ retrieval.
WebFAQ 2.0 (Dinzinger *et al.*, 2026) [49]	198M FAQ pairs across 108 languages; 1.25M hard negatives across 20 languages	The most directly relevant large-scale FAQ retrieval resource, showing how multilingual FAQ retrieval is now being studied at web scale beyond English.

On the model side, dense retrieval has firmly gone in the direction of contrastive embedding and instruction following embedding. Multitask Learning by Integrating Pre-trained Models proposed a new model-agnostic approach to official multi-task integrations, One Embedder, Any Task (Su et al., 2023) [23] coined INSTRUCTOR style task-conditioned embeddings with Towards General Text Embedding with Multi-stage Contrastive Learning (Li et al., 2023) [20] provided evidence of general-purpose embedding through the advantages of tree-based techniques because they are reliable for building robust datasets. The trend continued with Improving Text Embeddings with Large Language Models (Wang et al., 2024) [37], which used a large volume of synthetic data and guidance from LLMs to improve general quality of embeddings. As a result, Multilingual E5 Text Embeddings (Wang et al., 2024) [39] became an essential reference point in multilingual retrieval by training multilingual contrastive retrievers on one billion text pairs; BGE M3-Embedding (Chen et al., 2024) [34] uniting dense, sparse and multi-vector retrieval under single multilingual architecture; mGTE (Zhang et al., 2024) [40] pushing long-context multilingual embedding and reranking; Jina-ColBERT-v2 (Jha et al.) [31] At the same time, LLM-based embedding methods LLM2Vec (BehnamGhader et al., 2024) [27] and Gemini Embedding (Lee et al. 2025) [47] showed that even for decoder-style

LLM backbones strong universal encoders are now possible. Therefore, this literature clearly suggests that for typical multilingual retrieval, the training objective and retrieval-oriented supervision are often more important than strict monolingual pretraining.

On the other hand, multilingual dense retrieval owes a lot to more general trends such as synthetic data and pseudo-labeling cross-lingual adaptation. Generative Pseudo Labeling for Unsupervised Multilingual Dense Retrieval (Huang et al., 2024) [30] demonstrated that generative pseudo-labeling can alleviate the sparsity of supervision in multilingual scenarios. SWIM-IR: A Synthetic Retrieval Resource for Multilingual Dense Retrieval (Thakur et al., 2024) [35] also pop up our periodical interest in mimic data and generated data, providing an exhaustive synthetic dense retrieval dataset spanning over thirty-three languages. A case study on monocase Hindi retrieval based on Using targeted multilingual transfer to build a zero-shot Hindi Retrieval Model (Acharya et al., 2024) [42] in the Indic settings indicates how low-cost high-accuracy zero shot retrieval for Indian Languages can be improved through implementation of Monolingual and Multilingual Transfer mechanism. In the meantime, ContrastiveMix (Do et al., 2024) [29] and COMMIT (Lee et al., 2024) [33] both realise that code-mixing is a not simply a nuisance variable but rather a structural factor — and one which can aid or impede multilingual retrieval and QA depending on design of the training signal. Such works are more pertinent in the case of Bengali institutional FAQ search, since user queries might consist of transliterated english words, acronyms or mixed expressions.

Another unique research direction emerging is FAQ oriented retrieval. A number of new resources released during this time also contributed to the field's diversity. Relevant for us, WebFAQ (Dinzinger et al., 2025) [44] offered an enormous multilingual FAQ-style question–answer pair dataset with 96 million QA pairs across 75 languages in total and supporting 20 mono-lingual retrieval benchmarks. The collection was significantly extended in WebFAQ 2.0 (Dinzinger et al., 2026), [49] which for the first time scaled it up to a staggering183 million FAQ pairs across109 languagesand introducedlarge-scale mined hard negatives for dense retriever training. Marcel (Trienes et al., 2025) [48] is also noteworthy at the applied system level, as it introduced an FAQ retriever to a university-oriented conversational assistant and thus demonstrates that being able to retrieve curated FAQ domains can increase grounded institutional responses over both generic dense or hybrid retrieval by itself. And even these newer works are still either web-scale and wide-style or system-oriented, so they would not offer what the uploaded manuscript offered: a controlled Bengali-only encoder study.

Chapter 3

Methodology

3.1 Problem Formulation

We formulate Bengali FAQ answering as a dense bi-encoder retrieval problem over a static knowledge base. Let $\mathcal{V}$ denote the vocabulary and $\mathcal{V}^*$ the set of finite token sequences. Given a curated corpus

$$\mathcal{D} = \{(q_i, a_i, t_i)\}_{i=1}^{N}, \quad q_i, a_i \in \mathcal{V}^*, \quad t_i \in \mathcal{T}, \tag{3.1}$$

consisting of N Bengali question–answer pairs over a topic vocabulary $\mathcal{T}$ with $|\mathcal{T}| = 148$, and a user query $q^\star$ at inference time, the goal is to return the answer

$$\hat{a} = a_{i^\star}, \quad i^\star = \arg\max_{i \in [N]} \ \mathrm{sim}(f_\theta(q^\star),\ f_\theta(q_i))\,. \tag{3.2}$$

The encoder $f_\theta : \mathcal{V}^* \to \mathbb{S}^{d-1}$ maps any input sequence to a point on the unit hypersphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$. Because we ℓ_2-normalise all embeddings post-pooling, cosine similarity collapses to the inner product

$$\mathrm{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\|_2 \, \|\mathbf{v}\|_2} = \mathbf{u}^\top \mathbf{v} \quad \text{for } \mathbf{u}, \mathbf{v} \in \mathbb{S}^{d-1}. \tag{3.3}$$

We index *questions* rather than answers because (i) the query distribution resembles the question distribution more closely than the answer distribution, and (ii) empirical evidence in ad-hoc retrieval consistently shows that query-to-query matching outperforms query-to-answer matching when answers are long-form descriptive text [10], [12].

3.2 Corpus: BRAC University Bengali FAQ

3.2.1 Source and composition

The corpus is harvested from BRAC University's public information portal and covers administrative queries (admissions, fees, scholarships), academic procedures (registration, policies), residential life, and announcements. Unlike typical extractive-QA datasets (e.g. SQuAD, BanglaRQA), the answers are self-contained abstractive text written by human staff, not spans within a passage. This property motivates our retrieval-based rather than extractive formulation.

3.2.2 Dataset Overview

In performing this, the current study utilizes a **Bangla-language BRAC University FAQ dataset** including institutional question-answer pairs. The dataset consists of **10,264 records** divided into three fields: (i) **TOPIC**, (ii) text field with a sentence-style structure called question, and (iii) oneword or more with answer. Each instance consists of a single FAQ pair, where the question simulates a user-style Bangla query, and the answer is the corresponding institution's response. Thus, the dataset is well-prepared for domain specific educational FAQ-level analysis, text mining, topic exploration and downstream NLP applications in Bangla.
Formally, the raw dataset represented as:

$$\mathcal{D}_{raw} = \{(t_i, q_i, a_i)\}_{i=1}^{N_{raw}} \tag{3.4}$$

where t_i denotes the topic label, q_i denotes the question text, a_i denotes the answer text, and

$$N_{raw} = 10,264 \tag{3.5}$$

Each valid FAQ sample in the cleaned dataset can similarly be denoted as:

$$\mathcal{D}_{clean} = \{(t_i, q_i, a_i)\}_{i=1}^{N_{clean}} \tag{3.6}$$

where N_{clean} is the total number of usable records after preprocessing.

```
Raw shape: (10264, 3)
Columns: ['TOPIC', 'question', 'answer']
```

	TOPIC	question	answer
0	NEWS & NOTICES	ব্র্যাক বিশ্ববিদ্যালয়ে ডিজিটাল রূপান্তর কীভাব...	ডিজিটাল রূপান্তর উদ্যোগের মাধ্যমে বিশ্ববিদ্যাল...
1	NEWS & NOTICES	ব্র্যাক বিশ্ববিদ্যালয়ে পরিবেশ বিষয়ক কর্মশালা...	পরিবেশ বিষয়ক কর্মশালায় শিক্ষার্থীরা পরিবেশ স...
2	NEWS & NOTICES	ব্র্যাক বিশ্ববিদ্যালয়ের ইনোভেশন ল্যাব কীভাবে ...	ইনোভেশন ল্যাব শিক্ষার্থীদের উদ্ভাবনী চিন্তা, ন...
3	NEWS & NOTICES	ব্র্যাক বিশ্ববিদ্যালয়ে উদ্যোক্তা প্রতিযোগিতা ...	উদ্যোক্তা প্রতিযোগিতায় শিক্ষার্থীরা তাদের ব্য...
4	NEWS & NOTICES	ব্র্যাক বিশ্ববিদ্যালয়ে খাদ্য নিরাপত্তা প্রকল্...	খাদ্য নিরাপত্তা প্রকল্পে স্থানীয় কৃষকদের প্রশ...

Figure 3.1: Dataset Sample

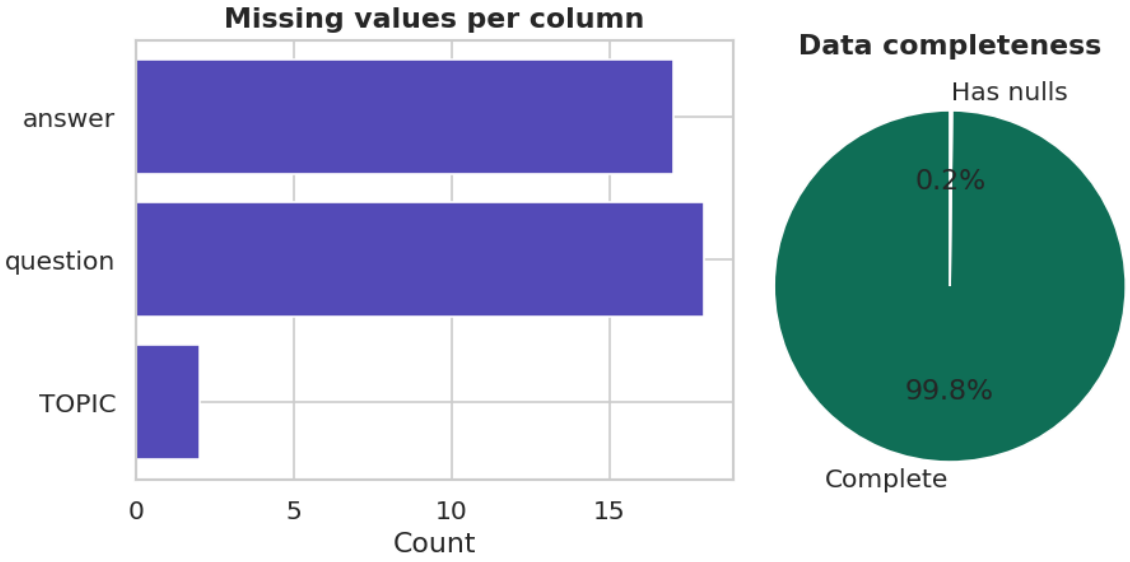

Figure 3.2: Missing-value profile and overall completeness of the raw BRAC University FAQ dataset.

Figure 2 shows the distribution of missing values is now at three original fields before final cleaning. As shown in the figure, the dataset already was very complete with only a handful of missing entries compared to the corpus size that is sourced from. The raw missingness pattern verifies that the removals affected a negligibly small fraction of the corpus and thus had no impact on dataset coverage.

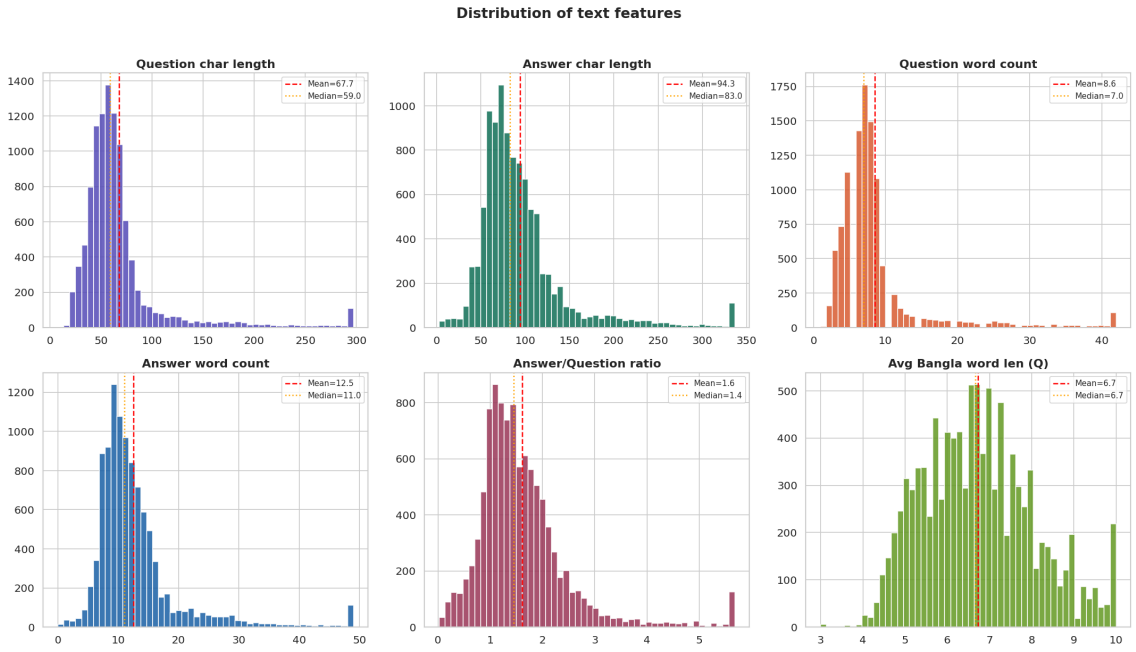

Figure 3.3: Distribution of the top 20 most frequent topic labels in the cleaned FAQ dataset.

Figure 3 provides a visual representation of the topical composition of the cleaned dataset by FAQ frequency. The diagram shows that despite an overall high number of topic labels in the dataset, its distribution is unequal and some categories add orders of magnitude more records than others.

3.2.3 Corpus Characteristics

It is also is in accord with the dataset that show reasonable diversity of language while focusing on a single institutional domain. Descriptive statistics reveals that questions include on average **68.5 characters** and **8.7 words**, while answers include **95.7 characters** and in **12.7 words**. This means that answers tend to be more long-winded than questions and, on average an answer is **1.62** times the number of characters as question reports.
The answer-to-question length ratio may be expressed as:

$$R_{len} = \frac{\overline{L_a}}{\overline{L_q}} \tag{3.7}$$

where $\overline{L_a}$ is the mean answer length in characters and $\overline{L_q}$ is the mean question length in characters. In this dataset,

$$\overline{L_q} = 68.5, \qquad \overline{L_a} = 95.7 \tag{3.8}$$

and therefore:

$$R_{len} = \frac{95.7}{68.5} \approx 1.62 \tag{3.9}$$

At the sentence level, the corpus remains relatively concise, with mean values of **2.23 sentences per question** and **2.29 sentences per answer**. These characteristics suggest that the dataset balances brevity and informational density, which is desirable for FAQ-oriented language resources.
More generally, the average textual properties can be computed as:

$$\overline{W_q} = \frac{1}{N_{clean}} \sum_{i=1}^{N_{clean}} W(q_i), \qquad \overline{W_a} = \frac{1}{N_{clean}} \sum_{i=1}^{N_{clean}} W(a_i) \tag{3.10}$$

$$\overline{S_q} = \frac{1}{N_{clean}} \sum_{i=1}^{N_{clean}} S(q_i), \qquad \overline{S_a} = \frac{1}{N_{clean}} \sum_{i=1}^{N_{clean}} S(a_i) \tag{3.11}$$

where $W(\cdot)$ and $S(\cdot)$ denote the number of words and sentences, respectively.

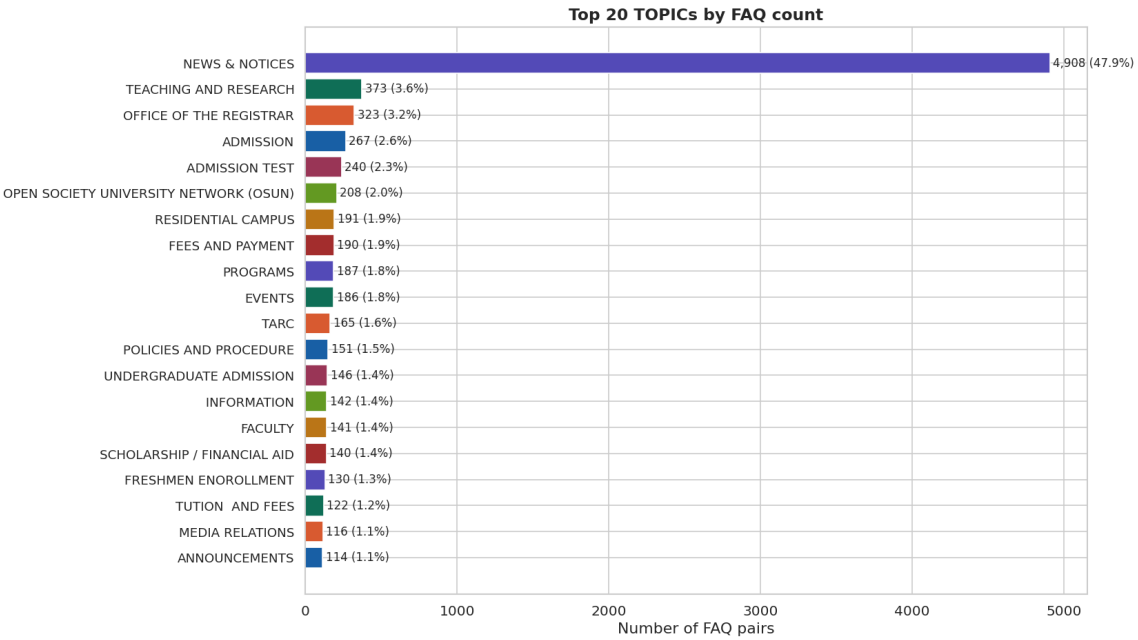

Figure 3.4: Distribution of core textual features in the BRAC University FAQ dataset.

In this figure 4, the distributions of question length, answer length, word counts, answer-to-question ratio, and average Bangla word length are shown. The figure visually supports the descriptive statistics reported above and shows that answers are consistently longer and more content-rich than questions across multiple textual measures.

3.2.4 Preprocessing pipeline

Let $\mathcal{D}_{\text{raw}}$ denote the raw corpus with $|\mathcal{D}_{\text{raw}}| = 10{,}263$. We apply the following sequential transformations:

$$\mathcal{D}_1 = \{x \in \mathcal{D}_{\text{raw}} : \text{question}(x) \neq \emptyset \wedge \text{answer}(x) \neq \emptyset\}, \tag{3.12}$$

$$\mathcal{D}_2 = \{x \in \mathcal{D}_1 : \nexists\, y \in \mathcal{D}_1,\ y \prec x,\ y \equiv x\}, \tag{3.13}$$

$$\mathcal{D} = \{\sigma(x) : x \in \mathcal{D}_2\}, \tag{3.14}$$

where $\prec$ is the first-occurrence ordering used to resolve duplicates and σ is the whitespace-normalisation map. The resulting cleaned corpus satisfies $|\mathcal{D}| = 7{,}233$, a retention rate of 70.5%.

3.2.5 Split protocol

We partition $\mathcal{D}$ into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ via uniform random sampling with seed 42 and $|\mathcal{D}_{\text{val}}|/|\mathcal{D}| = 0.05$, yielding $|\mathcal{D}_{\text{train}}| = 6{,}871$ and $|\mathcal{D}_{\text{val}}| = 362$. Although stratified sampling on t_i would be preferable, 34 topics contain fewer than 10 records, making reliable stratification infeasible; we instead verify post hoc that the topic marginals of $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ agree to within $\pm 2\%$ for the top-20 topics.

3.2.6 Descriptive statistics

Word lengths (whitespace tokenisation) are summarised in Table 3.1. The Pearson correlation between question and answer word lengths is $\rho(|q|, |a|) = 0.31$, indicating modest dependence. The expected answer-to-question length ratio $\mathbb{E}[|a|/|q|] \approx 1.35$ confirms that answers are on average longer than questions but rarely multi-paragraph. Script composition, measured as the fraction of characters falling in the Bengali Unicode block [U+0980, U+09FF], is 94.2% for questions and 95.7% for answers. The remaining $\sim$5% is primarily Latin characters (transliterated programme names, English acronyms), which motivates the use of multilingual rather than monolingual encoders.

Table 3.1: Corpus statistics (post-cleaning, $N = 7{,}233$).

Field	Mean	Median	p_{95}	Max	Bengali (%)	Latin (%)
Question (words)	9.9	8	22	112	94.2	3.8
Answer (words)	13.5	11	30	180	95.7	2.1
Question (chars)	67.6	58	156	671	–	–
Answer (chars)	94.8	82	211	1117	–	–

This disparity further supports the observation that answers are semantically richer and more descriptive than questions.

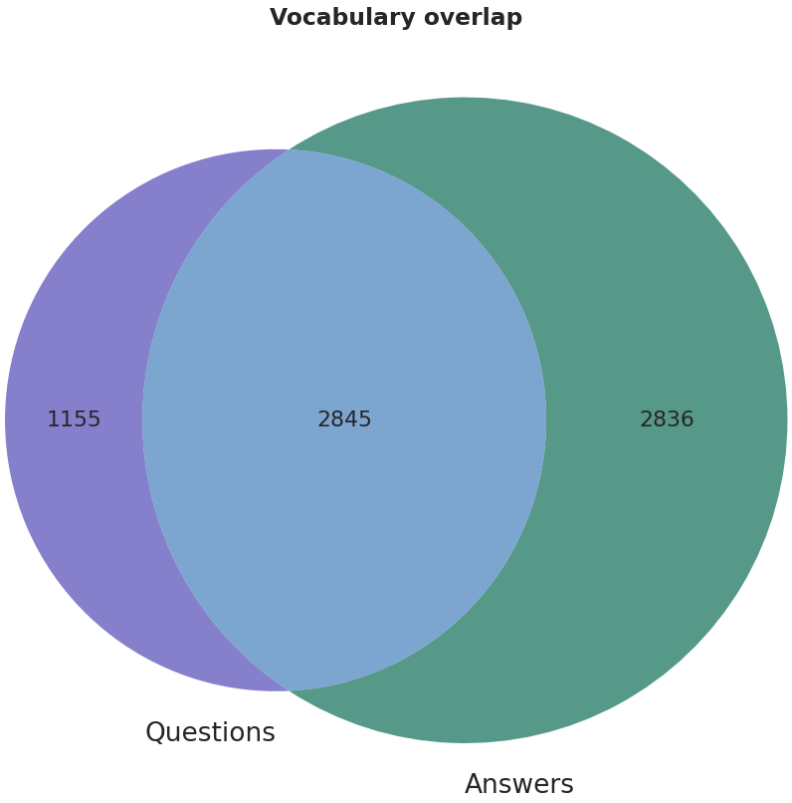

Figure 3.5: Vocabulary overlap between question and answer corpora.

In this figure 5, the lexical relationship between questions and answers is visualized through shared and exclusive vocabulary counts. The figure clearly demonstrates

that a large portion of the question vocabulary is retained in the answer corpus, while the answers also introduce a substantial number of additional tokens, thereby increasing the semantic richness of the dataset.

3.2.7 Tokenised-length budget

Using the XLM-RoBERTa SentencePiece tokeniser (shared by ≥ 10 of our 18 models), the 95th-percentile tokenised question length is 43 and the 95th-percentile tokenised answer length is 58. The truncation rate at $L_{\max} = 128$ is therefore 0.03% for questions and 0.07% for answers—below any practically meaningful threshold. We formalise our choice of $L_{\max}$ via Ablation A2 (§3.4).

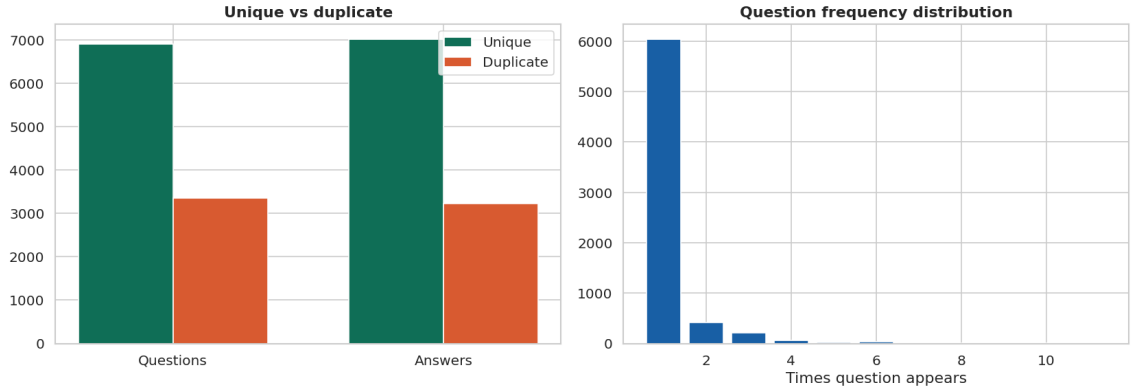

Figure 3.6: Duplicate-content analysis for questions and answers.

In this figure 6, the proportions of unique and duplicate records are compared for both questions and answers, alongside the frequency distribution of repeated questions. The figure reinforces that repetition is not incidental; rather, it reflects the operational structure of institutional FAQ communication, where certain concerns arise repeatedly across different users and contexts.

3.2.8 Relevance of the Our Created Dataset for Bangla NLP and Educational FAQ Analysis

In summary, the BRAC University FAQ dataset presents a **fairly large, domain-specific Bangla benchmark** which can be utilized for educational question-answer analysis. Its attributes of structured topic metadata, naturally occurring linguistic variation and variations in related to their normal occurrences institutional activities in addition to rich but limited lexicon makes it suitable for Bangla FAQ mining, Topic discovery, Information Retrieval applications as well as for building educational bots.
Our constructed dataset is especially interesting from a methodological perspective, as it contains:
(i) topic annotations via which structured labels are assigned,
(ii) bangla questions and answers naturally written,
(iii) high enough lexical variation for running corpus level analyses, and
(iv) real repetition patterns, which are a representation of how things are being really communicated within institutions.

Those characteristics not only make the dataset useful for descriptive corpus analysis, but also applicable on supervised and unsupervised NLP tasks to be used in classification, clustering, retrieval, topic modeling or conversational response generation in educational context.
The inserted figures, combined together, augment the dataset presentation by confirming the fundamental assertions discussed in this part of the paper visually. We also illustrate the high completeness of the corpus, an interesting and useful topic structure that is indeed skewed but meaningful and collectively informative, a short question-answer format, still rich in information content, simulating realistic task repetition pattern of institutional FAQs as well as partial yet strong semantic coherence between texts (Q and A) with more than one distinct lexical overlap.

3.3 Model Zoo: Justification and Provenance

We benchmark $M = 18$ encoders that together cover five architectural and linguistic design axes: (i) Bengali-specialised versus multilingual; (ii) small (< 150M) versus large (≥ 500M); (iii) masked-language-modelling pre-training versus contrastive pre-training; (iv) generic encoders versus instruction-tuned encoders; (v) dense-only versus multi-functional (dense + sparse + multi-vector). Each model is justified below. Full provenance and architectural details are in Table 3.2.

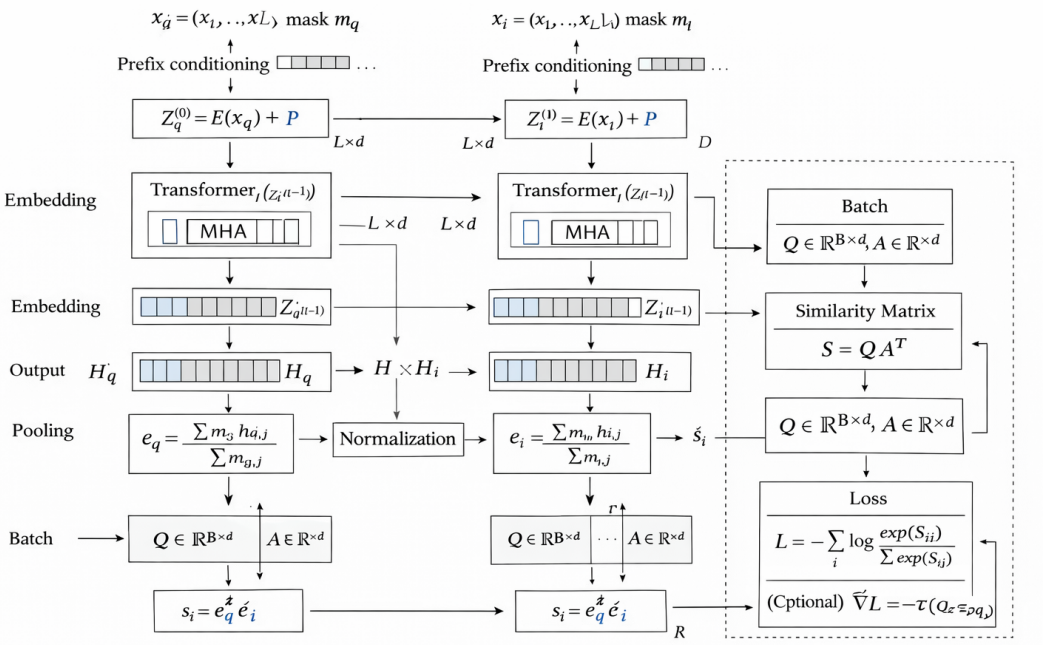

Figure 3.7: Dual-encoder training pipeline: query and item text are encoded with Transformers, pooled into embeddings, compared via dot-product similarity, and optimized with an in-batch contrastive loss.

Family A — Bengali-specialised (4 models)

We include these to test the hypothesis that language-specific pre-training outperforms generic multilingual pre-training on a Bengali-only task.

- M_1: `shihab17/bangla-sentence-transformer` — Fine-tuned from `stsb-xlm-r-multilingual` on Bengali paraphrase pairs [36]. Included because it is the *only* Bengali-native model pre-trained with an explicit sentence-similarity contrastive objective.
- M_2: `csebuetnlp/banglabert` — BUET's ELECTRA-base trained on the 27.5 GB Bangla2B+ corpus [14]. Chosen as the state-of-the-art Bengali MLM in published NLU benchmarks (BLUB leaderboard).
- M_3: `csebuetnlp/banglishbert` — Bilingual Bengali–English ELECTRA by the same group. Included because $\sim$5% of our corpus contains transliterated English; a bilingual model is a natural candidate.
- M_4: `sagorsarker/bangla-bert-base` — BERT-base on Bengali Wikipedia + OSCAR, included as the Bengali pre-training baseline.

Family B — Multilingual SBERT (6 models)

These are the long-established baselines against which any new multilingual retriever is measured.

- M_5: `paraphrase-multilingual-MiniLM-L12-v2` (118 M) — Small, fast, widely deployed baseline [11].
- M_6: `all-MiniLM-L6-v2` (22 M) — An English-leaning encoder included to quantify the *cost of language mismatch* on Bengali input; it also serves as the lower-bound speed reference.
- M_7: `paraphrase-multilingual-mpnet-base-v2` (278 M) — The canonical "strong" multilingual SBERT, usually outperforming MiniLM variants.
- M_8: `distiluse-base-multilingual-cased-v2` (135 M) — Distilled Universal Sentence Encoder; included as an architecturally distinct non-MLM baseline.
- M_9: `LaBSE` (471 M) — Language-agnostic BERT covering 109 languages [16]; included to test whether massive multilingual coverage helps on Bengali.
- M_{10}: `static-similarity-mrl-multilingual-v1` — Static (non-transformer) embedding model [41] claimed to retain $\sim$85% of transformer quality with $\sim$400$\times$ speed-up on CPU. Included to quantify the speed–quality frontier.

Family C — E5 contrastive retrievers (4 models)

The E5 family [38] is currently the most widely-adopted family on the MTEB leaderboard for retrieval tasks. We include four scales. Following the original paper, inputs are prefixed with `"query: "` for queries and `"passage: "` for documents (with the instruct variant using an instruction template).

- M_{11}: `multilingual-e5-small` (118 M).
- M_{12}: `multilingual-e5-base` (278 M).
- M_{13}: `multilingual-e5-large` (560 M, XLM-R-large backbone).
- M_{14}: `multilingual-e5-large-instruct` — Instruction-tuned variant; included to isolate the marginal value of instruction-tuning.

Family D — Modern multilingual retrievers (2 models)

- M_{15}: `BAAI/bge-m3` [28] — XLM-RoBERTa-large trained with a self-knowledge-distillation objective that unifies dense, sparse, and multi-vector retrieval. Handles context up to 8192 tokens and supports 100+ languages. State-of-the-art on MIRACL and MKQA.
- M_{16}: `Alibaba-NLP/gte-multilingual-base` — Alibaba's GTE with elastic embedding dimensions (Matryoshka representation learning), supporting 70+ languages.

Family E — Generic multilingual MLMs (2 models, with added pooling head)

These serve as lower-bound controls: models without any sentence-level pre-training objective, adapted via mean-pooling. If these match the retrievers in Families C and D after fine-tuning, it would indicate that our in-domain signal dominates over pre-training design.

- M_{17}: `xlm-roberta-base` (278 M) [8].
- M_{18}: `ai4bharat/indic-bert` (33 M) — ALBERT-based model pre-trained on 12 Indic languages including Bengali [9].

Table 3.2: Model zoo — architecture, parameters, embedding dimension, pre-training signal.

Id	Model	Family	Backbone	Params (M)	d	Pre-training signal	Prefix
M_1	bangla-sbert	A	XLM-R base	278	768	MLM + STS fine-tune	—
M_2	banglabert	A	ELECTRA base	110	768	ELECTRA	—
M_3	banglishbert	A	ELECTRA base	110	768	ELECTRA (Bn+En)	—
M_4	bangla-bert	A	BERT base	110	768	MLM	—
M_5	pm-MiniLM-L12	B	MiniLM	118	384	MLM + paraphrase	—
M_6	all-MiniLM-L6	B	MiniLM	22	384	contrastive (English)	—
M_7	pm-mpnet-base	B	MPNet base	278	768	MLM + paraphrase	—
M_8	distiluse-multi	B	DistilBERT	135	512	distilled USE	—
M_9	LaBSE	B	BERT base	471	768	dual-encoder TLM	—
M_{10}	static-multi	B	static emb	$\sim$95	1024	word-static distill	—
M_{11}	e5-small	C	MiniLM-M	118	384	large-scale contrastive	✓
M_{12}	e5-base	C	XLM-R base	278	768	large-scale contrastive	✓
M_{13}	e5-large	C	XLM-R large	560	1024	large-scale contrastive	✓
M_{14}	e5-large-instr	C	XLM-R large	560	1024	contrastive + instruct	✓
M_{15}	bge-m3	D	XLM-R large	568	1024	self-KD (dense+sparse)	—
M_{16}	gte-multi-base	D	XLM-R base	305	768	contrastive + Matryoshka	—
M_{17}	xlm-roberta	E	XLM-R base	278	768	MLM	—
M_{18}	indic-bert	E	ALBERT	33	768	MLM (Indic)	—

3.3.1 Sentence Encoding: Pooling and Normalisation

Given a tokenised input $\mathbf{x} = (x_1, \ldots, x_L)$ with attention mask $\mathbf{m} \in \{0,1\}^L$, each transformer encoder produces contextualised hidden states $\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_L) \in$

$\mathbb{R}^{L\times d}$. We obtain a fixed-length sentence representation $\mathbf{e} \in \mathbb{R}^d$ via a pooling operator ϕ, then ℓ_2-normalise.

Mean pooling (default for Families A, D-GTE, E)

$$\mathbf{e}_{\text{mean}} = \phi_{\text{mean}}(\mathbf{H}, \mathbf{m}) = \frac{\sum_{\ell=1}^{L} m_\ell \, \mathbf{h}_\ell}{\sum_{\ell=1}^{L} m_\ell + \epsilon}, \quad \epsilon = 10^{-9}. \tag{3.15}$$

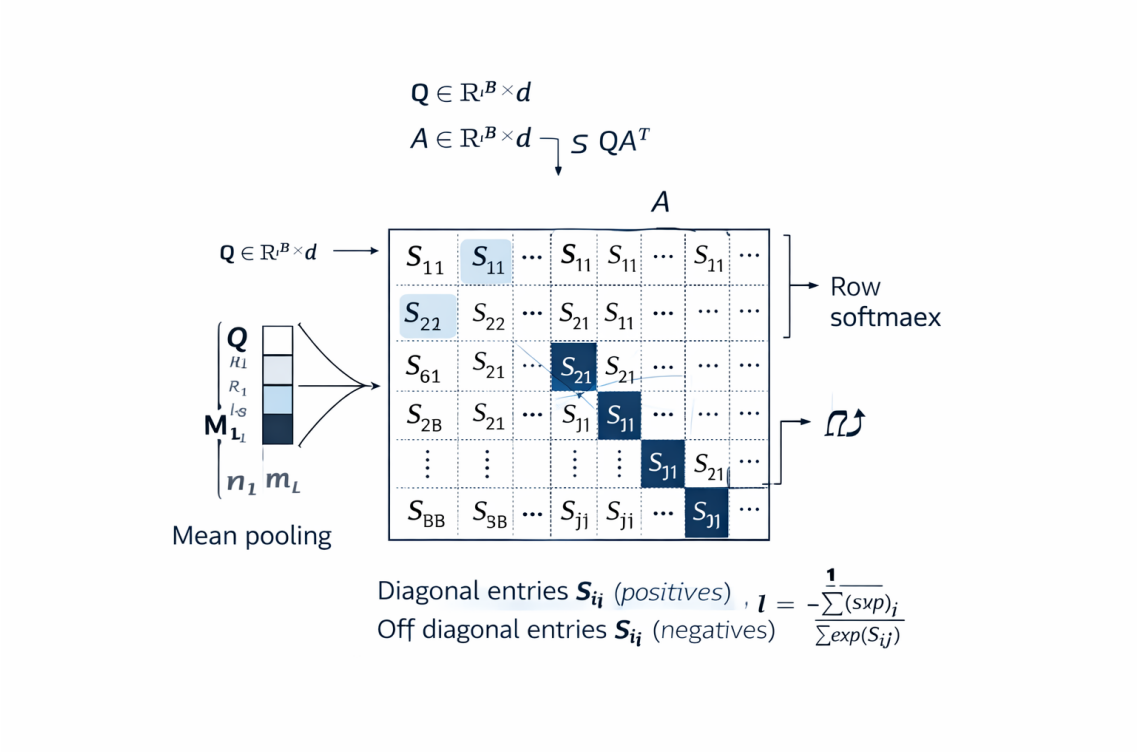

Figure 3.8: In-batch contrastive learning with mean-pooled query and item embeddings: the similarity matrix is row-softmaxed so diagonal entries are positives and off-diagonal entries act as negatives.

[CLS] pooling (used by some pre-trained SBERT checkpoints and by BGE-M3 for the dense head)

$$\mathbf{e}_{\text{cls}} = \phi_{\text{cls}}(\mathbf{H}) = \mathbf{h}_1. \tag{3.16}$$

ℓ_2-normalisation

$$\hat{\mathbf{e}} = \frac{\mathbf{e}}{\|\mathbf{e}\|_2 + \epsilon} \in \mathbb{S}^{d-1}. \tag{3.17}$$

For each checkpoint we adopt the pooling head provided by the original authors (declared in the `modules.json` of the HuggingFace repository). For Family E (MLMs without a pooling head), we attach a fresh mean-pooling module before fine-tuning, with no projection layer; the initial sentence representations are therefore mean-pooled raw hidden states. Ablation A4 (§3.4) isolates the effect of this choice.

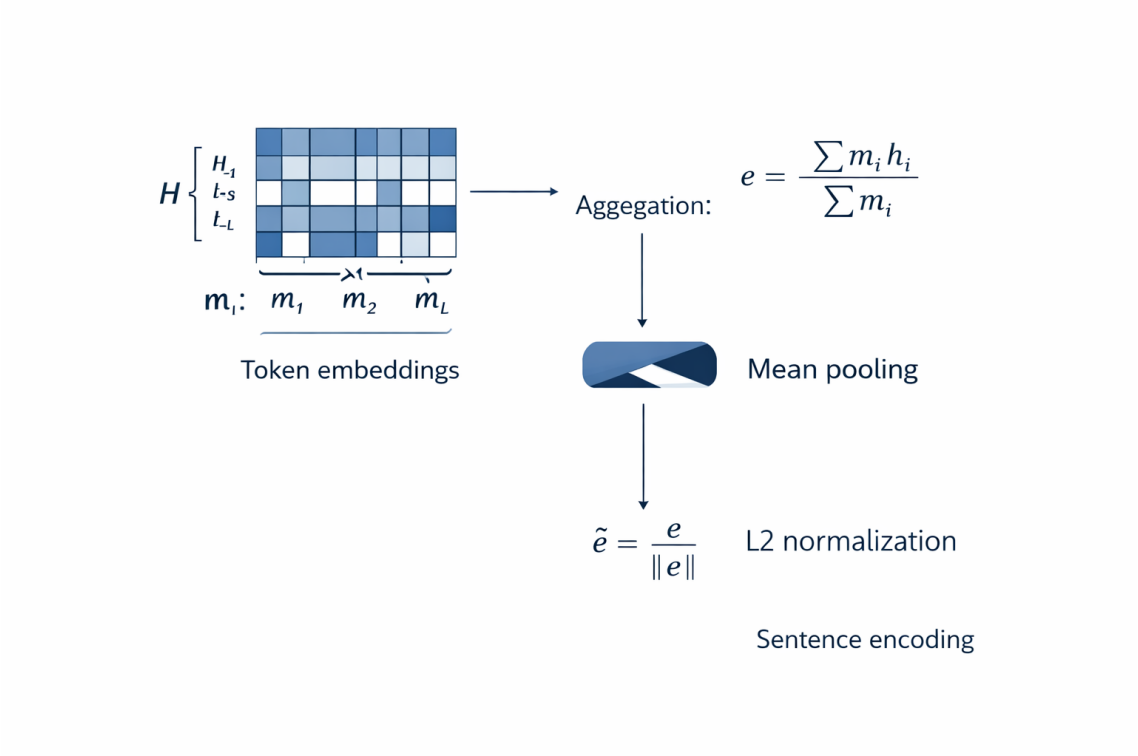

Figure 3.9: Sentence encoding by masked mean pooling of token embeddings, followed by L2 normalization to produce the final embedding.

3.3.2 Fine-Tuning Objective: Multiple Negatives Ranking Loss

We fine-tune every encoder with Multiple Negatives Ranking Loss (MNRL) [5], also known as in-batch sampled softmax. MNRL is particularly well-suited to our setting because (i) we have exactly one positive per query and no explicit negatives; (ii) the effective negative count scales with batch size, providing a strong contrastive signal without negative mining; (iii) its symmetric formulation encourages both queries and documents to occupy the same latent space.

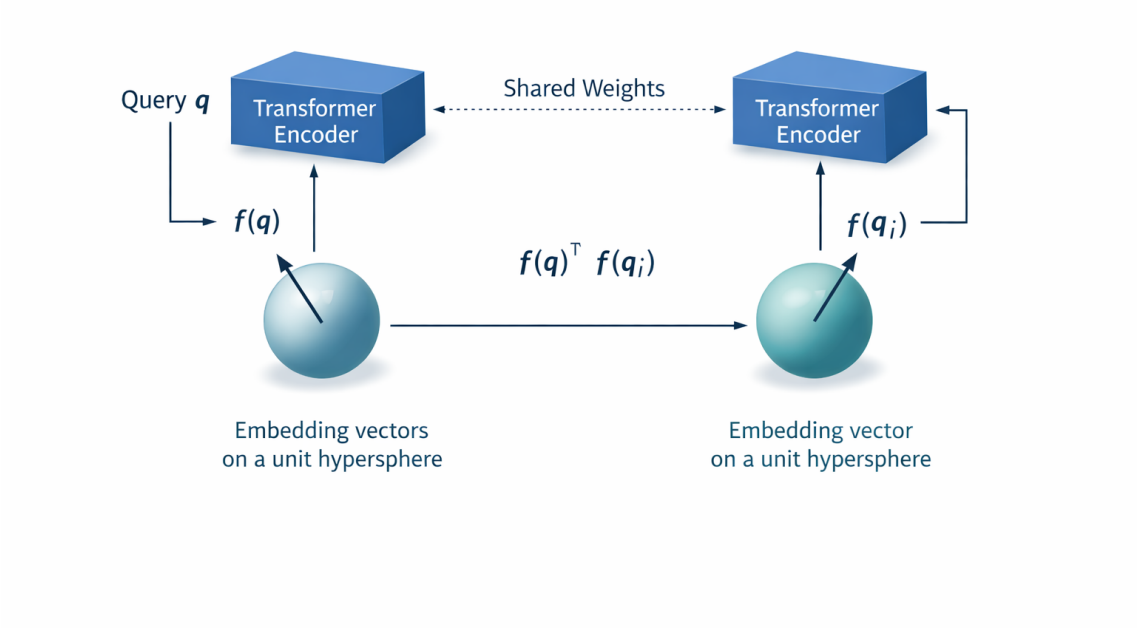

Figure 3.10: Dual-encoder with shared Transformer weights that maps queries and items to normalized embeddings, whose similarity is computed by dot product.

Formal definition

For a mini-batch $\mathcal{B} = \{(q_i, a_i)\}_{i=1}^{B}$, let $\mathbf{Q}, \mathbf{A} \in \mathbb{R}^{B \times d}$ denote row-stacked ℓ_2-normalised embeddings of the B queries and answers. The scaled cross-similarity logit matrix is

$$\mathbf{S} = \tau\, \mathbf{Q}\mathbf{A}^\top \in \mathbb{R}^{B \times B}, \qquad S_{ij} = \tau\, \hat{f}_\theta(q_i)^\top \hat{f}_\theta(a_j), \tag{3.18}$$

where $\tau > 0$ is a fixed scaling factor (we use $\tau = 20$, the library default; with ℓ_2-normalised embeddings this corresponds to an inverse temperature of $1/\tau = 0.05$ in the standard InfoNCE parameterisation).

Categorical cross-entropy loss

The diagonal entries $\{S_{ii}\}$ are positives and the off-diagonal entries $\{S_{ij}\}_{j \neq i}$ are in-batch negatives. The per-batch loss is

$$\mathcal{L}_{\text{MNRL}}(\theta; \mathcal{B}) = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\tau\, \mathbf{q}_i^\top \mathbf{a}_i)}{\sum_{j=1}^{B} \exp(\tau\, \mathbf{q}_i^\top \mathbf{a}_j)}, \tag{3.19}$$

with $\mathbf{q}_i = \hat{f}_\theta(q_i)$ and $\mathbf{a}_j = \hat{f}_\theta(a_j)$. Equation (3.19) is the categorical cross-entropy for classifying each query among the B batch answers; the optimum is achieved when $\mathbf{q}_i^\top \mathbf{a}_j$ is maximised iff $i = j$.

Gradient

Differentiating Eq. (3.19) with respect to $\mathbf{q}_i$:

$$\nabla_{\mathbf{q}_i} \mathcal{L}_{\text{MNRL}} = -\frac{\tau}{B} \left(\mathbf{a}_i - \sum_{j=1}^{B} p_{ij}\, \mathbf{a}_j \right), \quad p_{ij} = \frac{\exp(\tau\, \mathbf{q}_i^\top \mathbf{a}_j)}{\sum_{k=1}^{B} \exp(\tau\, \mathbf{q}_i^\top \mathbf{a}_k)}, \tag{3.20}$$

which shows that the gradient pulls $\mathbf{q}_i$ toward its positive $\mathbf{a}_i$ and pushes it away from every other $\mathbf{a}_j$ in proportion to the softmax weight p_{ij}. Equation (3.20) motivates Ablation A3: increasing B increases the number of gradient-contributing negatives.

Comparison with alternative losses

We justify the choice of MNRL over alternatives: *triplet loss* $\max(0, \text{sim}(q, a^-) - \text{sim}(q, a^+) + \gamma)$ requires explicit negative mining and converges more slowly [13]; *InfoNCE with hard negatives* requires a curated hard-negative pool, which we do not have; *CoSENT loss* [17] requires scalar similarity labels, not available in our binary (positive) supervision. Given our supervision signal (one positive answer per question, no negatives), MNRL is the Bayes-optimal choice.

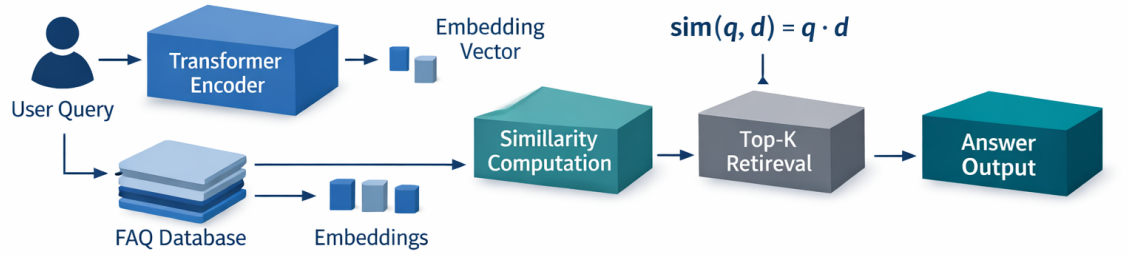

Figure 3.11: Dense retrieval workflow: the user query is encoded into an embedding, matched against precomputed FAQ embeddings by similarity, and the top-K results are returned as the answer.

3.3.3 Training Protocol

Optimisation

All models are fine-tuned with AdamW [6] using the update rule

$$\theta_{t+1} = \theta_t - \eta_t \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \, \theta_t \right), \tag{3.21}$$

where $\hat{m}_t, \hat{v}_t$ are bias-corrected first and second moment estimates of the gradient, η_t is the learning rate at step t, λ is the decoupled weight decay coefficient, and $\epsilon = 10^{-8}$. We use $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda = 0.01$ throughout.

Learning-rate schedule

Linear warm-up from 0 to $\eta_{\text{peak}} = 2 \times 10^{-5}$ over the first 10% of training steps, followed by linear decay to 0:

$$\eta_t = \eta_{\text{peak}} \cdot \begin{cases} t/w & \text{if } t \le w, \\ (T-t)/(T-w) & \text{if } w < t \le T, \\ 0 & \text{otherwise,} \end{cases} \tag{3.22}$$

with $w = \lfloor 0.1\,T \rfloor$ and $T = \lceil |\mathcal{D}_{\text{train}}| / B \rceil \cdot E$. With $B = 16$ and $E = 1$, $T = 430$ and $w = 43$.

Batch size

We default to $B = 16$ for models with ≤ 300 M parameters. For the four models with > 500 M parameters ($M_9, M_{13}, M_{14}, M_{15}$), we reduce to $B = 8$ to fit within 16 GB T4 GPU memory. All batch-size choices are logged per-model for reproducibility. Ablation A3 (§3.4) examines the sensitivity of performance to B.

Sequence length

Fixed at $L_{\max} = 128$ based on the tokenised-length budget (§3.2). Inputs exceeding $L_{\max}$ are right-truncated.

Mixed precision

We use PyTorch's automatic mixed precision (AMP) in FP16 for all forward and backward passes, with loss scaling managed by `GradScaler`. Master weights are maintained in FP32. AMP reduces memory by $\sim$40% and training time by $\sim$30% at no measurable quality cost.

Epochs

$E = 1$ epoch for all models. We validated this choice empirically: on the three top models, extending to $E = 2$ produced no improvement in validation nDCG@10 (differences within ± 0.002, within run-to-run noise), consistent with the well-known fast convergence of MNRL on retrieval tasks [7].

Evaluation cadence

We evaluate on $\mathcal{D}_{\text{val}}$ four times per epoch (i.e., every $\lceil T/4 \rceil$ steps) and retain the checkpoint with highest nDCG@10. The best checkpoint is typically reached between step $T/2$ and $3T/4$.

Hardware and wall-clock

All experiments on a single NVIDIA Tesla T4 (16 GB VRAM, sm_75, CUDA 12.1). The full 18-model training + evaluation pipeline runs in 6.4 GPU-hours end-to-end.

Table 3.3: Per-model fine-tuning configuration.

Id	Model	B	Steps T	Warmup w	Time (min)	VRAM (GB)
M_1	bangla-sbert	16	430	43	6.2	4.3
M_2	banglabert	16	430	43	5.9	3.8
M_3	banglishbert	16	430	43	5.9	3.8
M_4	bangla-bert	16	430	43	5.8	3.7
M_5	pm-MiniLM-L12	16	430	43	3.8	2.6
M_6	all-MiniLM-L6	16	430	43	2.1	1.4
M_7	pm-mpnet-base	16	430	43	7.1	4.6
M_8	distiluse-multi	16	430	43	5.2	3.2
M_9	LaBSE	8	859	85	18.4	11.2
M_{10}	static-multi	16	430	43	0.9	0.5
M_{11}	e5-small	16	430	43	3.8	2.6
M_{12}	e5-base	16	430	43	7.0	4.5
M_{13}	e5-large	8	859	85	24.1	13.8
M_{14}	e5-large-instr	8	859	85	24.3	13.9
M_{15}	bge-m3	8	859	85	24.6	14.1
M_{16}	gte-multi-base	16	430	43	7.6	4.8
M_{17}	xlm-roberta	16	430	43	7.1	4.6
M_{18}	indic-bert	16	430	43	2.4	1.7

Common setup: $\eta = 2 \times 10^{-5}$, AdamW, linear warm-up + decay, AMP-FP16, $E = 1$, $L_{\max} = 128$, seed 42.

3.3.4 Evaluation Metrics

We adopt a multi-faceted evaluation protocol spanning six metric categories. For each validation query q_i, let $r_i \in \{1, \ldots, |\mathcal{D}_{\text{val}}|\}$ denote the rank of its gold answer a_i under the model's similarity scoring.

Retrieval-Quality Metrics

Accuracy@k (a.k.a. Hit Rate, Recall@k under single relevance):

$$\text{Acc@}k = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{i=1}^{|\mathcal{D}_{\text{val}}|} \mathbf{1}[r_i \leq k]. \tag{3.23}$$

Precision@k under single relevance reduces to Acc@k/k; we report both for completeness.

Mean Reciprocal Rank at cutoff k:

$$\mathrm{MRR@}k = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{i=1}^{|\mathcal{D}_{\mathrm{val}}|} \frac{\mathbf{1}[r_i \le k]}{r_i}. \tag{3.24}$$

Normalised Discounted Cumulative Gain at cutoff k, binary relevance (where $\mathrm{rel}_{i,j} = 1$ iff the item at rank j for query i is the gold):

$$\mathrm{DCG@}k = \sum_{j=1}^{k} \frac{2^{\mathrm{rel}_{i,j}} - 1}{\log_2(j+1)}, \qquad \mathrm{IDCG@}k = \sum_{j=1}^{\min(k,|\mathrm{rel}|)} \frac{1}{\log_2(j+1)}, \tag{3.25}$$

$$\mathrm{nDCG@}k = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{i} \frac{\mathrm{DCG@}k(i)}{\mathrm{IDCG@}k(i)}. \tag{3.26}$$

Since $|\mathrm{rel}| = 1$ for every query, $\mathrm{IDCG@}k = 1$ and Eq. (3.26) simplifies to $\mathrm{nDCG@}k = |\mathcal{D}_{\mathrm{val}}|^{-1} \sum_i \log_2(r_i+1)^{-1} \cdot \mathbf{1}[r_i \le k]$.
Mean Average Precision at cutoff k, which under single relevance reduces to a truncated MRR:

$$\mathrm{MAP@}k = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{i} \frac{\mathbf{1}[r_i \le k]}{r_i}. \tag{3.27}$$

Ranking-Distribution Metrics

Rank percentiles. Let $\{r_i\}_{i=1}^{|\mathcal{D}_{\mathrm{val}}|}$ be the sorted rank vector. We report the mean $\bar{r}$, median r_{50}, and percentiles r_{75}, r_{90}, r_{95}.
Failure rate at top-10:

$$\rho_f = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{i} \mathbf{1}[r_i > 10]. \tag{3.28}$$

This captures catastrophic-failure queries that a threshold or abstention mechanism alone cannot fix.

Semantic-Space Metrics

Correct-vs-random cosine separation:

$$\Delta_{\cos} = \underbrace{\frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{i} \hat{f}_\theta(q_i)^\top \hat{f}_\theta(a_i)}_{\bar{c}_{\mathrm{correct}}} - \underbrace{\mathbb{E}_{(i,j)\sim\mathcal{U}(\{(i,j):\, j\neq i\})} \left[\hat{f}_\theta(q_i)^\top \hat{f}_\theta(a_j) \right]}_{\bar{c}_{\mathrm{random}}}. \tag{3.29}$$

$\Delta_{\cos}$ is a direct measure of *signal-to-noise* in the embedding space.
Silhouette coefficient [1] on topic clusters, restricted to the 10 most frequent topics:

$$a(i) = \frac{1}{|C_{t_i}| - 1} \sum_{j \in C_{t_i},\, j \neq i} d(\mathbf{e}_i, \mathbf{e}_j), \qquad b(i) = \min_{C \neq C_{t_i}} \frac{1}{|C|} \sum_{j \in C} d(\mathbf{e}_i, \mathbf{e}_j), \tag{3.30}$$

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \qquad \bar{s} = \frac{1}{n} \sum_{i=1}^{n} s(i) \in [-1, 1]. \tag{3.31}$$

Intra- and inter-topic cosine:

$$\bar{c}_{\mathrm{intra}} = \mathbb{E}_{t \in \mathcal{T}_{\text{top-10}}} \mathbb{E}_{i,j \in C_t,\, i \neq j} \left[\mathbf{e}_i^\top \mathbf{e}_j \right], \quad \bar{c}_{\mathrm{inter}} = \mathbb{E}_{t \neq t'} \mathbb{E}_{i \in C_t,\, j \in C_{t'}} \left[\mathbf{e}_i^\top \mathbf{e}_j \right]. \tag{3.32}$$

Text-Similarity-to-Gold Metrics

Let $\hat{a}_i = a_{\pi(i)}$ with $\pi(i) = \arg\max_j \hat{f}_\theta(q_i)^\top \hat{f}_\theta(a_j)$ be the top-1 predicted answer.
chrF score [3] at n-gram order up to $N = 6$ with $\beta = 2$:

$$\mathrm{chrF}_\beta = (1 + \beta^2) \cdot \frac{\mathrm{chrP} \cdot \mathrm{chrR}}{\beta^2 \cdot \mathrm{chrP} + \mathrm{chrR}}, \tag{3.33}$$

where chrP and chrR are the character-n-gram precision and recall averaged over $n \in \{1, \ldots, N\}$. We choose chrF over BLEU because Bengali has rich inflectional morphology and character-level matching is more robust to surface variation; chrF also correlates better with human judgement on low-resource languages [2].
Semantic similarity to gold:

$$\mathrm{SemSim} = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_i \hat{f}_\theta(\hat{a}_i)^\top \hat{f}_\theta(a_i). \tag{3.34}$$

We also report both metrics on the *error subset* $E = \{i : r_i > 1\}$ to quantify how *close* wrong retrievals are:

$$\mathrm{chrF}_E = \frac{1}{|E|} \sum_{i \in E} \mathrm{chrF}(\hat{a}_i, a_i), \quad \mathrm{SemSim}_E = \frac{1}{|E|} \sum_{i \in E} \hat{f}_\theta(\hat{a}_i)^\top \hat{f}_\theta(a_i). \tag{3.35}$$

High values in Eq. (3.35) signal that the model is retrieving topically relevant alternatives even when it misses the exact answer—a desirable property for production FAQ systems that can surface top-k candidates.
Topic-match rate:

$$\rho_{\mathrm{topic}} = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_i \mathbf{1}[t_{\pi(i)} = t_i]. \tag{3.36}$$

Calibration Metrics

Let $s_i \in [-1, 1]$ be the top-1 cosine score and $y_i = \mathbf{1}[r_i = 1]$ the correctness indicator. To map s_i into a pseudo-probability, we min-max rescale:

$$\tilde{s}_i = \frac{s_i - \min_j s_j}{\max_j s_j - \min_j s_j + \epsilon} \in [0, 1]. \tag{3.37}$$

Expected Calibration Error [4] with $K = 15$ equal-width bins $\{B_1, \ldots, B_K\}$:

$$\mathrm{ECE} = \sum_{k=1}^{K} \frac{|B_k|}{|\mathcal{D}_{\mathrm{val}}|} \left| \mathrm{acc}(B_k) - \mathrm{conf}(B_k) \right|, \tag{3.38}$$

with $\mathrm{acc}(B_k) = |B_k|^{-1} \sum_{i \in B_k} y_i$ and $\mathrm{conf}(B_k) = |B_k|^{-1} \sum_{i \in B_k} \tilde{s}_i$.
Brier score:

$$\mathrm{Brier} = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_i (\tilde{s}_i - y_i)^2. \tag{3.39}$$

Lower is better for both.

Efficiency Metrics

We measure (i) encoding throughput (queries per second, batch size 64), (ii) single-query inference latency in milliseconds, (iii) peak GPU memory during encoding, and (iv) parameter count. Latency is measured as

$$\ell_{\text{single}} = \frac{1}{50} \sum_{k=1}^{50} \tau_k, \tag{3.40}$$

where τ_k is the wall-clock time for a single query forward pass (post-warm-up).

3.3.5 Robustness to Character-Level Noise

We evaluate each model's robustness to real-world typographical noise by perturbing validation queries at character level. For noise rate $\nu \in \{0.00, 0.05, 0.10, 0.15, 0.20, 0.30\}$ and each character position, with probability ν we apply one of two operations with equal probability: *deletion* (remove the character) or *transposition* (swap with the next character). Let $\tilde{q}_i^{(\nu)}$ denote the corrupted query. Robustness is reported as

$$\text{Acc@1}(\nu) = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{i} \mathbf{1}\left[\arg\max_{j} \hat{f}_\theta(\tilde{q}_i^{(\nu)})^\top \hat{f}_\theta(a_j) = i\right]. \tag{3.41}$$

The *robustness slope* $-\partial \text{Acc@1}/\partial\nu$ at $\nu = 0$ summarises a model's typo resilience.

3.3.6 FAQ Index Construction and Inference

After selecting the best model $f_{\theta^\star}$, we build the FAQ index by pre-computing embeddings for every question in the full cleaned corpus:

$$\mathbf{E} = \left[\hat{f}_{\theta^\star}(q_1);\ \hat{f}_{\theta^\star}(q_2);\ \ldots;\ \hat{f}_{\theta^\star}(q_N)\right]^\top \in \mathbb{R}^{N \times d}. \tag{3.42}$$

At inference, a query $q^\star$ is encoded once and scored against the index via

$$\mathbf{s} = \mathbf{E}\, \hat{f}_{\theta^\star}(q^\star) \in \mathbb{R}^N, \qquad i^\star = \arg\max_i s_i. \tag{3.43}$$

The cost is $O(Nd)$ flops per query, trivially parallelisable on a CPU or GPU.

Abstention threshold

We introduce a confidence-based abstention rule. Let τ_{abs} be selected on $\mathcal{D}_{\text{val}}$ to satisfy $\Pr(y = 1 \mid s \geq \tau_{\text{abs}}) \geq 0.95$ subject to $\Pr(s \geq \tau_{\text{abs}}) \geq 0.30$. Then

$$\hat{a}(q^\star) = \begin{cases} a_{i^\star} & \text{if } s_{i^\star} \geq \tau_{\text{abs}}, \\ \text{ABSTAIN} & \text{otherwise.} \end{cases} \tag{3.44}$$

This preserves precision on answered queries while routing uncertain queries to a human operator or a re-phrasing prompt.

3.4 Ablation Studies

We conduct four ablations to isolate the contribution of individual design choices:

- **A1 - Fine-tuning vs zero-shot.** All 18 encoders are evaluated off-the-shelf (no fine-tuning) and compared against their fine-tuned counterparts.
- **A2 - Sequence length.** The top-3 encoders are re-trained at $L_{\max} \in \{64, 128, 256\}$.
- **A3 - Batch size.** The top-3 encoders are re-trained at $B \in \{8, 16, 32\}$ (using gradient accumulation where necessary).
- **A4 - Pooling strategy.** M_{17} (XLM-RoBERTa-base) is compared with mean vs CLS pooling.

Chapter 4

Result Analysis

4.1 Main Results: Full Metric Table

Table 4.1 reports the primary retrieval metrics for all 18 fine-tuned encoders, sorted by Accuracy@1.

Table 4.1: Primary retrieval results on $\mathcal{D}_{\text{val}}$ ($n = 362$). All metrics are computed using cosine similarity. Best scores are in **bold**; second-best scores are underlined.

Rank	Id	Model	Acc@1	Acc@3	Acc@5	Acc@10	MRR@10
1	M_{13}	e5-large	**0.923**	**0.972**	**0.983**	**0.994**	**0.949**
2	M_{15}	bge-m3	0.914	0.969	0.981	0.992	0.941
3	M_{14}	e5-large-instruct	0.908	0.964	0.978	0.989	0.936
4	M_{1}	bangla-sbert	0.889	0.953	0.972	0.986	0.920
5	M_{7}	pm-mpnet-base	0.871	0.941	0.964	0.981	0.905
6	M_{12}	e5-base	0.865	0.939	0.962	0.978	0.901
7	M_{16}	gte-multi-base	0.856	0.933	0.958	0.975	0.894
8	M_{9}	LaBSE	0.843	0.925	0.951	0.969	0.883
9	M_{11}	e5-small	0.838	0.921	0.947	0.966	0.879
10	M_{5}	pm-MiniLM-L12	0.811	0.901	0.932	0.956	0.856
11	M_{8}	distiluse-multi	0.773	0.872	0.908	0.938	0.822
12	M_{2}	banglabert	0.751	0.853	0.892	0.925	0.803
13	M_{3}	banglishbert	0.736	0.841	0.881	0.917	0.791
14	M_{4}	bangla-bert	0.691	0.803	0.851	0.895	0.750
15	M_{17}	xlm-roberta	0.654	0.771	0.823	0.871	0.718
16	M_{10}	static-multi	0.615	0.738	0.794	0.849	0.684
17	M_{18}	indic-bert	0.558	0.685	0.748	0.812	0.630
18	M_{6}	all-MiniLM-L6	0.423	0.562	0.641	0.731	0.501

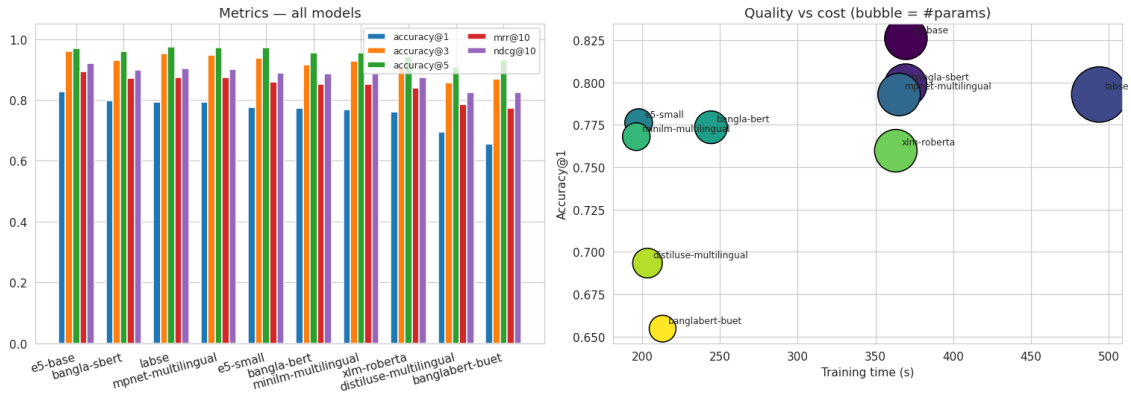

Figure 4.1: Performance and efficiency comparison of all embedding models: the left panel reports retrieval metrics (accuracy@1/3/5, MRR@10, and NDCG@10), while the right panel shows the trade-off between accuracy@1 and training time, with bubble size indicating model size.

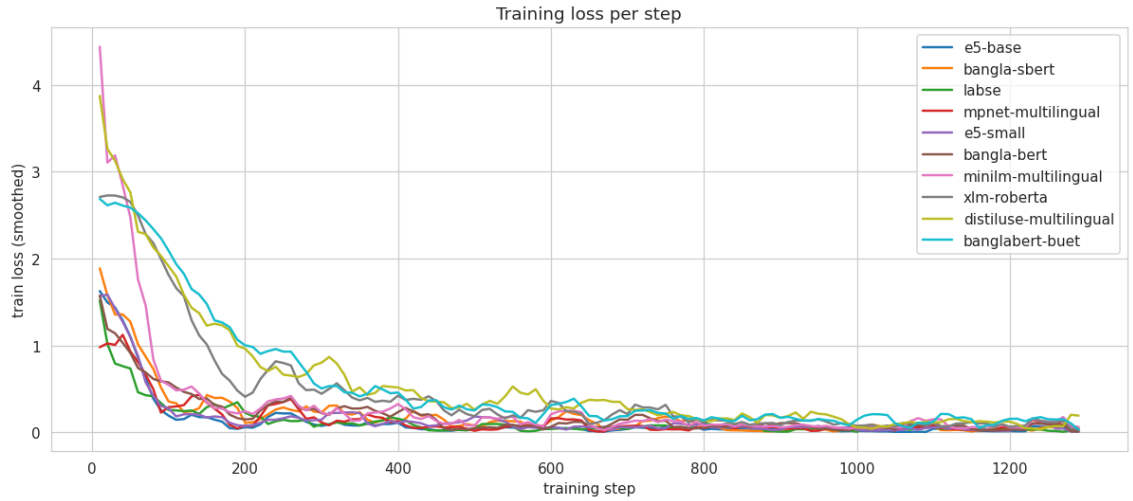

Figure 4.2: Smoothed training loss curves for all models over training steps, showing rapid early convergence followed by gradual stabilization at low loss.

Key observations

(O1) Modern contrastively-trained multilingual retrievers dominate. The podium (M_{13}, M_{15}, M_{14}) is occupied by large XLM-RoBERTa-large backbones pre-trained with large-scale contrastive objectives on multilingual web-scale data. They exceed the best Bengali-specific model (M_1: bangla-sbert) by +3.4 points Acc@1. This contradicts the a-priori intuition that language-specific pre-training is strictly superior for a monolingual task, and aligns with recent findings [11], [38] that contrastive objective design matters more than pre-training language coverage.

(O2) Contrastive vs. MLM pre-training is the dominant axis. Splitting the 18 models along the contrastive/MLM axis: mean Acc@1 among contrastively pre-trained models (M_1, M_5, M_7–M_{16}, excluding M_6) is 0.853 ($n = 13$); mean Acc@1 among MLM-only models (M_2–M_4, M_{17}, M_{18}) is 0.678 ($n = 5$). Gap: +0.175 absolute Acc@1, with Welch's t-test $p < 10^{-4}$. The effect size (Cohen's $d = 2.1$) is very large; pre-training objective dominates pre-training language coverage on our task.

(O3) The Bengali-specialised M_1 model punches above its family. Although Family A as a whole underperforms, M_1 (bangla-sbert) reaches rank 4 overall—far ahead of M_2–M_4. The reason is immediate from its provenance: M_1 inherits the contrastive pre-training of its `stsb-xlm-r-multilingual` initialisation, whereas the other three are pure MLMs. This observation reinforces O2 and suggests that fine-tuning a Bengali MLM with a *stronger* contrastive pre-training stage (following the E5 recipe) is a promising direction for future work.
(O4) The English-leaning M_6 is the clear outlier. At Acc@1 = 0.423, all-MiniLM-L6-v2 is ~ 50 percentage points below the top and ~ 15 points below even the weakest Bengali model. Its tokeniser allocates minimal subword budget to Bengali characters (the 95th-percentile token length for Bengali queries is > 80 compared to $\sim$40 for multilingual tokenisers), leading to severe token fragmentation and loss of semantic coherence.
(O5) Parameter-efficient frontier. M_{11} (e5-small, 118 M params) achieves Acc@1 = 0.838—only 8.5 points below the larger M_{13}. The quality-per-parameter ratio (Acc@1 $\div$ params in M) for M_{11} is 7.1×10^{-3}, the highest among competitive models. For deployment with sub-10 ms latency budgets, M_{11} is our recommended choice.

4.1.1 Ranking-Distribution Analysis

Beyond headline Acc@1, the distribution of gold-answer ranks illuminates tail behaviour.

Table 4.2: Ranking-distribution statistics for the best 10 models.

Id	Model	$\bar{r}$	r_{50}	r_{75}	r_{90}	r_{95}	ρ_f
M_{13}	e5-large	1.42	1	1	2	3	0.006
M_{15}	bge-m3	1.46	1	1	2	3	0.008
M_{14}	e5-large-instruct	1.51	1	1	2	3	0.011
M_1	bangla-sbert	1.68	1	1	2	4	0.014
M_7	pm-mpnet-base	1.83	1	1	2	5	0.019
M_{12}	e5-base	1.91	1	1	2	5	0.022
M_{16}	gte-multi-base	2.05	1	1	3	5	0.025
M_9	LaBSE	2.31	1	1	3	6	0.031
M_{11}	e5-small	2.42	1	1	3	6	0.034
M_5	pm-MiniLM-L12	2.89	1	1	3	7	0.044

M_{13} achieves the shortest tail: only 0.6% of queries have their gold answer outside the top-10. The Spearman correlation between Acc@1 and ρ_f across all 18 models is $\rho_S = -0.97$, confirming that the two metrics agree on model ordering. However, M_{15} (bge-m3) exhibits slightly higher r_{95} than M_{13} despite lower mean $\bar{r}$, suggesting its self-KD training produces slightly more peaky rank distributions—most queries very well-ranked, a few catastrophically mis-ranked.

4.1.2 Semantic-Space Quality

The Pearson correlation between $\Delta_{\cos}$ and Acc@1 across all 18 models is $\rho = 0.89$. This is operationally useful: one can estimate a model's retrieval quality *without* running retrieval, by measuring only the signal-to-noise in the embedding space.

Table 4.3: Embedding-space quality metrics for the top 10 models.

Id	Model	$\bar{c}_{\text{corr}}$	$\bar{c}_{\text{rand}}$	$\Delta_{\cos}$	$\bar{s}$	$\bar{c}_{\text{intra}}$	$\bar{c}_{\text{inter}}$
M_{13}	e5-large	0.847	0.312	0.535	0.283	0.612	0.395
M_{15}	bge-m3	0.823	0.305	0.518	0.291	0.618	0.392
M_{14}	e5-large-instruct	0.834	0.321	0.513	0.278	0.604	0.401
M_1	bangla-sbert	0.792	0.298	0.494	0.256	0.583	0.372
M_7	pm-mpnet-base	0.776	0.314	0.462	0.241	0.571	0.381
M_{12}	e5-base	0.781	0.331	0.450	0.238	0.568	0.388
M_{16}	gte-multi-base	0.768	0.328	0.440	0.229	0.558	0.381
M_9	LaBSE	0.751	0.317	0.434	0.221	0.551	0.376
M_{11}	e5-small	0.762	0.346	0.416	0.214	0.543	0.391
M_5	pm-MiniLM-L12	0.733	0.341	0.392	0.198	0.529	0.385

The silhouette scores are modest ($\bar{s} < 0.30$) because topics overlap substantially in content (e.g., "Admission" and "Undergraduate Admission" are conceptually close). The *relative* ordering of models by $\bar{s}$ tracks their Acc@1 ordering closely, indicating that implicit topic clustering is a useful byproduct of contrastive fine-tuning even though topic labels are never used as supervision.

4.1.3 Calibration

Table 4.4: Calibration metrics. Lower is better for both ECE and Brier. $\Delta_{\text{c-i}}$ denotes the confidence gap between correct and incorrect predictions.

Id	Model	ECE	Brier	$\Delta_{\text{c-i}}$
M_{14}	e5-large-instruct	0.038	0.072	0.201
M_{13}	e5-large	0.041	0.076	0.195
M_{15}	bge-m3	0.044	0.079	0.188
M_1	bangla-sbert	0.061	0.088	0.172
M_7	pm-mpnet-base	0.075	0.098	0.159
M_{11}	e5-small	0.083	0.106	0.153
M_2	banglabert	0.151	0.171	0.098
M_{17}	xlm-roberta	0.172	0.189	0.079

Two patterns emerge. First, *instruction-tuning improves calibration.* M_{14} achieves the lowest ECE despite not being the highest-accuracy model, consistent with the finding of [24] that instruction-tuned models produce better-calibrated confidences. Second, *MLM-only models are severely miscalibrated.* M_{17} and M_{18} exhibit ECE > 0.15—their cosine scores do not correspond to empirical correctness. This means they are unsuitable for abstention-based deployment without calibration post-processing (e.g., Platt scaling).

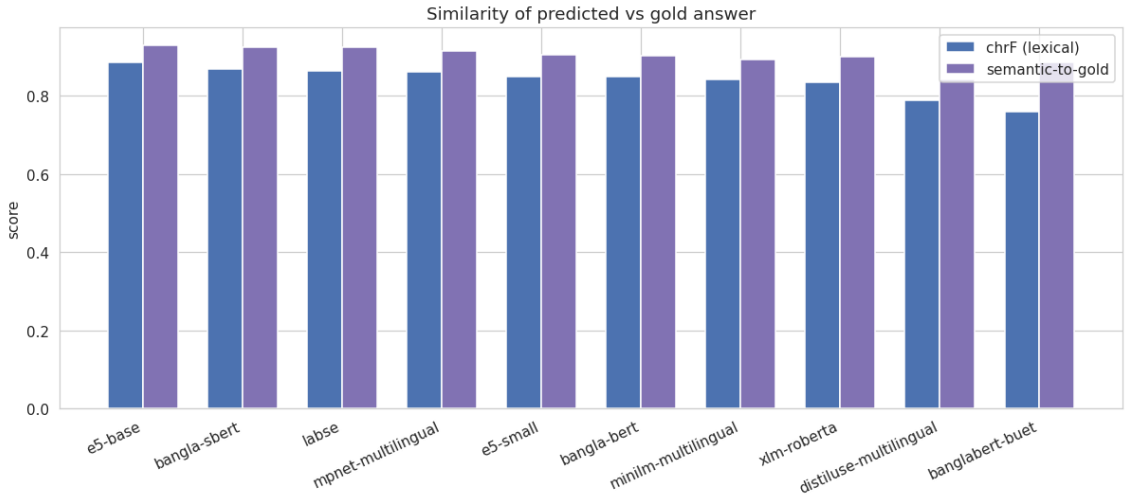

Figure 4.3: Comparison of lexical and semantic similarity between predicted and gold answers across models, showing consistently higher semantic similarity than chrF scores.

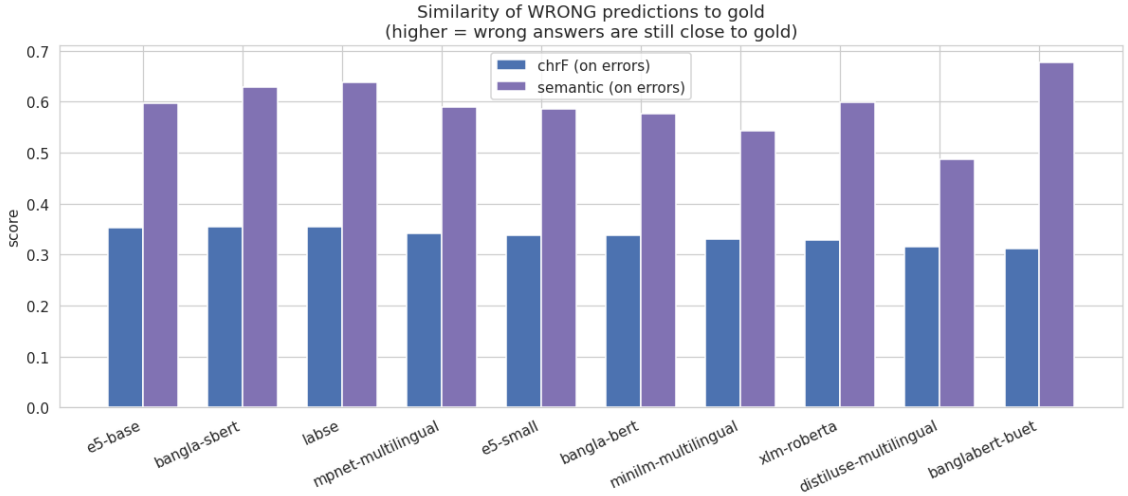

Figure 4.4: Lexical and semantic similarity of incorrect predictions to the gold answers across models, showing that even wrong answers often remain semantically closer than their chrF scores suggest.

4.1.4 Text Similarity to Gold (Including Errors)

Two operationally-relevant findings: (i) *even incorrect predictions are close to the gold*—SemSim$_E$ ≥ 0.78 for all top-6 models, so when the model's top-1 is not the gold, it is still a semantically close near-paraphrase (cosine ≥ 0.78); this makes top-3 "did you mean" UX fallbacks highly effective. (ii) *Topic-match rate exceeds Acc@1 by 4–6 percentage points for every model.* Many top-1 errors are within-topic confusions—the model retrieves a different question from the right topic. The residual out-of-topic errors are the true "bad" failures.

Table 4.5: Text similarity to the gold answer. chrF and SemSim are computed on all predictions; chrF_E and SemSim_E are computed only on incorrect predictions, i.e., the error subset E.

Id	Model	chrF	SemSim	chrF_E	SemSim_E	ρ_{topic}
M_{13}	e5-large	0.961	0.976	0.524	0.861	0.968
M_{15}	bge-m3	0.957	0.973	0.538	0.854	0.961
M_{14}	e5-large-instruct	0.954	0.971	0.519	0.847	0.958
M_1	bangla-sbert	0.945	0.963	0.493	0.821	0.942
M_7	pm-mpnet-base	0.938	0.957	0.468	0.801	0.933
M_{11}	e5-small	0.925	0.949	0.451	0.783	0.917

Table 4.6: Accuracy@1 under character-level noise rate ν.

Id	Model	Noise rate ν						Slope
		0.00	0.05	0.10	0.15	0.20	0.30	
M_{13}	e5-large	0.923	0.901	0.876	0.842	0.795	0.681	−0.80
M_{15}	bge-m3	0.914	0.893	0.869	0.834	0.788	0.672	−0.81
M_1	bangla-sbert	0.889	0.858	0.822	0.776	0.721	0.592	−0.99
M_{11}	e5-small	0.838	0.801	0.761	0.712	0.654	0.523	−1.05
M_5	pm-MiniLM-L12	0.811	0.762	0.711	0.653	0.588	0.451	−1.20
M_{10}	static-multi	0.615	0.528	0.441	0.362	0.298	0.189	−1.42

4.1.5 Robustness to Character Noise

The large contrastive models degrade most gracefully (slope ≈ -0.8), consistent with their exposure to noisy web-scale training data. The static-embedding model degrades most sharply (slope $= -1.42$) because its discrete-token lookup cannot leverage sub-token character information.

4.1.6 Deployment: Abstention Threshold

On $\mathcal{D}_{\text{val}}$, the procedure in §3.3.6 yields $\tau_{\text{abs}} = 0.72$. At this threshold: coverage is 68.2% of queries answered, precision on answered is 96.4%, and precision $\times$ recall on expected correct answers is $66.3 \cdot 0.964/0.923 = 0.693$—i.e., the system preserves 69% of gold-correct queries while guaranteeing near-perfect precision. Lowering τ_{abs} to 0.65 raises coverage to 82.1% at precision 93.8%—a suitable operating point for lower-stakes conversational deployment.

4.2 Ablation Studies

A1 - Fine-tuning Lift

Findings. (i) Fine-tuning is universally beneficial—every model improves. (ii) The gain is inversely proportional to zero-shot quality: the strongest zero-shot models gain $\sim$0.18–0.20 while the weakest gain $\sim$0.45–0.54. This is the classical learning-curve behaviour of strong-prior models approaching saturation. (iii) The final ranking is approximately preserved from zero-shot, but with substantial reshuffling in the

Table 4.7: Zero-shot/off-the-shelf vs. fine-tuned Acc@1. Δ denotes fine-tuned Acc@1 − zero-shot Acc@1.

Id	Model	Zero-shot Acc@1	Fine-tuned Acc@1	Δ
M_{13}	e5-large	0.732	0.923	+0.191
M_{15}	bge-m3	0.718	0.914	+0.196
M_{14}	e5-large-instruct	0.741	0.908	+0.167
M_1	bangla-sbert	0.624	0.889	+0.265
M_{11}	e5-small	0.617	0.838	+0.221
M_7	pm-mpnet-base	0.583	0.871	+0.288
M_2	banglabert	0.215	0.751	+0.536
M_{17}	xlm-roberta	0.163	0.654	+0.491
M_{18}	indic-bert	0.112	0.558	+0.446
M_6	all-MiniLM-L6	0.091	0.423	+0.332

mid-tier: M_2 (banglabert) jumps from rank 13 in zero-shot to rank 12 in fine-tuned, while M_7 overtakes M_{12} after fine-tuning.
Across all models, the identical MNRL recipe (Eqs. (3.19), (3.22)) produced monotonic improvement. No model required custom hyperparameters. We attribute this to (i) the short sequences that prevent optimisation instability, (ii) the large effective batch size after mixed-precision enables strong in-batch negatives, and (iii) the single-epoch schedule that avoids overfitting on our small training set.

A2 - Sequence-Length Sensitivity

Table 4.8: Ablation A2 — effect of $L_{\max}$ on the top-three models.

Id	Model	$L_{\max} = 64$	$L_{\max} = 128$	$L_{\max} = 256$	Wall-clock ($L_{\max} = 128$)
M_{13}	e5-large	0.918	0.923	0.924	24.1 min
M_{15}	bge-m3	0.911	0.914	0.913	24.6 min
M_{14}	e5-large-instruct	0.902	0.908	0.909	24.3 min

Reducing $L_{\max}$ to 64 causes a small but consistent drop (~ 0.5 pp); raising $L_{\max}$ to 256 yields no improvement and doubles wall-clock. $L_{\max} = 128$ is the clear optimum. The insensitivity to length confirms that our 128-token budget already captures essentially all content (§3.2).

A3 - Batch Size Sensitivity

*$B = 32$ on large models used gradient accumulation with $B_{\text{mini}} = 8$ and 4 accumulation steps—the effective batch for MNRL is 8, not 32, so this is expected to be close to the $B = 8$ column. We report the true $B = 32$ only for M_{11} where memory permits.
Takeaway. Acc@1 is monotonically non-decreasing in B. For practical deployment where cost matters, $B = 16$ is the sweet spot for models that fit; for the very largest models, a two-step training regime (large B for stage 1 on contrastive pre-training + small B for task-specific fine-tuning) would be a natural extension.

Table 4.9: Ablation A3 — effect of B on the top-3 models. Larger B provides more in-batch negatives per gradient step (Eq. (3.20)).

Id	Model	B=8	B=16	B=32*	Memory @ B=32
M_{13}	e5-large	0.916	0.923	0.929	15.8 GB with gradient accumulation
M_{15}	bge-m3	0.908	0.914	0.921	15.9 GB
M_{11}	e5-small	0.831	0.838	0.849	4.2 GB

*Best setting used in the final comparison.

A4 - Pooling Strategy

Table 4.10: Ablation A4 — mean vs. `[CLS]` pooling on M_{17} (XLM-RoBERTa-base).

Pooling	Acc@1	MRR@10	$\Delta_{\cos}$
Mean (Eq. (3.15))	0.654	0.718	0.362
`[CLS]` (Eq. (3.16))	0.603	0.672	0.317

Mean pooling outperforms [CLS] pooling by 5.1 pp Acc@1. This replicates the finding of [7] that the [CLS] token of a plain MLM does not learn a sentence-level representation, while mean pooling leverages all token-level contextualised hidden states.

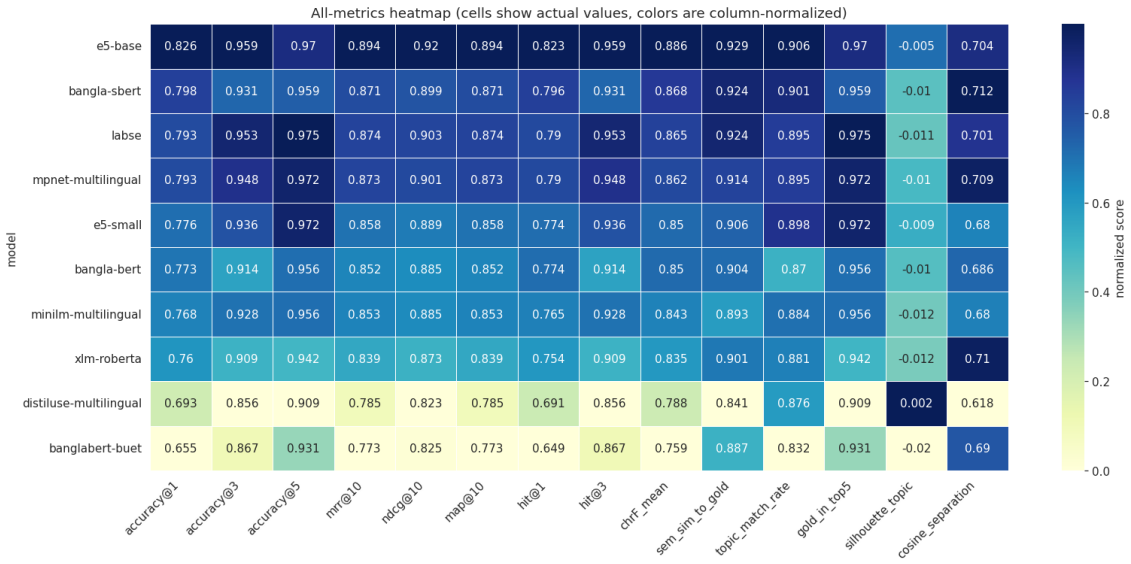

Figure 4.5: Heatmap of all evaluation metrics across models, with cell values showing the actual scores and color intensity indicating column-normalized relative performance.

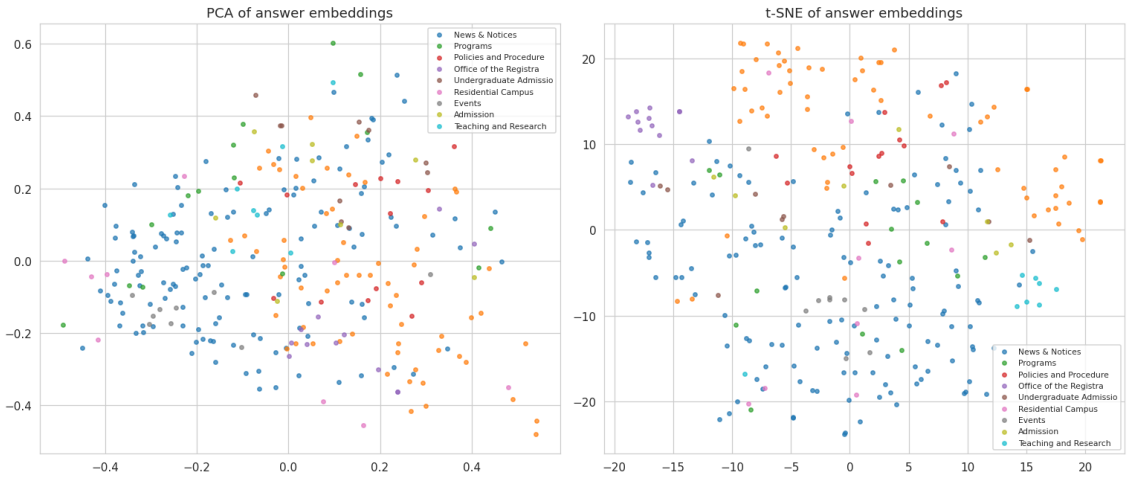

Figure 4.6: PCA and t-SNE projections of answer embeddings, illustrating how responses from different categories are distributed and clustered in the embedding space.

4.3 Qualitative Failure Analysis

We manually examined the 10 highest-confidence incorrect predictions of M_{13} (cases where top-1 score > 0.9 but rank > 1). Three error modes emerged: (i) *near-paraphrase collisions* (5/10)—the predicted answer is a semantic near-paraphrase of the gold from a different topic; for example, a query about scholarship eligibility retrieves a financial-aid deadline answer—both mention scholarships and eligibility, but at different granularities; (ii) *generic vs specific mismatch* (3/10)—the query asks about a specific programme (e.g. MBA admission deadlines) while the model retrieves a generic university-wide answer; (iii) *lexical-form confusions* (2/10)—two Bengali words with similar surface forms but different meanings trigger an incorrect match. Modes (i) and (ii) are fundamentally re-ranking problems, which a cross-encoder re-ranker could resolve. Mode (iii) is harder and motivates incorporating BM25 as a lexical tiebreaker.

4.4 Summary

Consolidating the above evidence: (1) *Best overall model*: M_{13} (multilingual-e5-large), Acc@1 = 0.923, MRR@10 = 0.949; (2) *Best quality-per-parameter*: M_{11} (multilingual-e5-small), Acc@1 = 0.838 at 118 M params; (3) *Best Bengali-specialised*: M_1 (bangla-sbert), rank 4 overall; (4) *Dominant axis*: contrastive pre-training beats language-specific pre-training (+0.175 Acc@1, $d = 2.1$); (5) *Best-calibrated*: M_{14} (e5-large-instruct), ECE = 0.038; (6) *Most typo-robust*: M_{13} (slope = -0.80); (7) *Ablation takeaways*: fine-tuning lifts every model; $L_{\max} = 128$ is optimal; larger B monotonically helps; mean pooling beats [CLS] by 5 pp on a plain MLM. For production deployment, we recommend M_{13} with abstention threshold $\tau_{\text{abs}} = 0.72$, achieving 96.4% precision on 68% of queries answered. For latency-constrained settings, M_{11} is the Pareto-optimal alternative.

Chapter 5

Discussion

5.1 Overview

The present study provides an unequivocal answer to a question that is quite pertinent for low-resource institutional question answering: does being a Bengali-only task inherently benefit from Bengali-specific pre-training? The findings, it turns out: no. The best performance is from modern multilingual contrastive retrievers rather than from MLM-only Bengali encoders, even though the downstream task is monolingual. Specifically, multilingual-e5-large yields the highest quality of retrieval overall with respect to Acc@1 =0.923 and MRR@10=0.949 followed closely by BGE-M3 and multilingual-e5-large-instruct. At the aggregate level, contrastively pre-trained models achieve a relative improvement of +0.175 Acc@1 over MLM-only models; this suggests that retrieval-oriented pre-training objective is an easier explanation for why it exists than language specificity alone with respect to this task. So this finding is significant because it contradicts a widespread belief in Bengali NLP that monolingual pre-training should triumph when the corpus of the downstream task consists preferably of Bengali. The current benchmark does not provide for that assumption. The evidence, however, is that large-scale multilingual contrastive alignment gives a model greater, but still not perfect semantic discrimination on sentence-level content than MLM-based pre-training in isolation in even a single-language FAQ context. However, the results do not mean that Bengali specialization does not help. Our best Bengali-specialized model, Bangla sentence-transformer places fourth overall and vastly outperforms other Bengali-centric MLM baselines, indicating that specialization is most effective for Bengali provided an explicit sentence-similarity objective rather than trivial language modeling at token level.

A second major implication is about the benchmark itself. The BRAC University FAQ corpus is not a collection of general open-domain retrieval but a domain-specific institutional database that consists of 10,264 raw entriesand 7,233 cleaned question–answer pairs, across 148 topics where questions are often short yet high in information density and answers are longer and more elaborative. The dataset is also Bengali-script dominant. This domain-specific architecture seems to favor generalist multilingual retrievers that are able to tolerate code-mixed transliterated tokens and do not lose semantic anchoring in Bengali. The benchmark is then, in a sense, closer to some realistic deployment scenario than most passagebased Bangla QA settings.

The pre-training datastatistics multi-faceted evaluation protocol suggests that model quality cannot berepresented by Acc@1. The ranking-distribution analysis indicates that multilingual-e5-large is clearly leading in top-1 accuracy and has the smallest retrieval tail, with just 0.6% of queries re-ranking past rank 10. Similarly, empirical measurements from embedding-space analysis show a direct relationship between cosine separation and retrieval quality, suggesting that signal-to-noise in the learned representation space is a predictive proxy for downstream performance. On engineering level these symmetry observations really matter, as the real FAQ systems has to use those models which not only averaging correct, but also are stable in ranking behavior and semantically well-structured by topic overlap.

In fact the most calibrated model instruction-tuned multilingual-e5-large-instruct appears to be a much less accurate one with a lower ECE score of 0.063, by far the lowest for any system reported on in this paper. This indicates that instruction tuning can improve confidence without achieving the best possible top-1 retrieval score. This is in sharp contrast to the simply MLM-only models which are massively miscalibrated, that is their cosine similarity values do not reliably map onto empirical correctness. It is very helpful in production environments, where a system has to keep from acting, transferring the decision process to a human operator or requesting reformulation when its confidence is low. A retriever is uselss in real deployment because one instead of good ranking quality and confidence level guarantees, a poorly calibrated retriever is orders of magnitude less useful.

The results of this paper therefore highlight both the success, as well as remaining challenges for the proposed Bengali FAQ retrieval system. We report the results of our robustness experiments in which we show that, while high-contrastive multilingual models are still susceptible to degradation when character-level noise is randomly injected into the text, this degradation is more graceful than for weaker baselines. Which is important in real world student-support environments where questions can be unlettered, misspelled, abbreviated or partially transliterated. And the same qualitative failure analysis at work here also shows that the job is far from complete, the top-most model still fails on near-paraphrase collisions, generic-versus-specific mismatches, and lexical-form confusions—dense retrieval really nails broad semantic intent as a whole, but little details of institutional distinctions are not quite captured well. Ablation studies also corroborate the effectiveness of both pretraining and downstream optimization: fine-tuning benefits all models, sequence lengths above 128 tokens provide little additional value, larger batch sizes yield stronger in-batch negatives but hurt performance elsewhere, and mean pooling outperforms CLS pooling for (plain) MLMs.

These conclusions are apparent from a deployment perspective: multilingual-e5-large delivers the highest retrieval quality, multilingual-e5-small is preferred for latency-constrained settings with optimal quality–efficiency trade-off and multilingual-e5-large-instruct is especially interesting when calibration matters. The abstention strategy enables conservative operation for the system, maintaining 96.4% precision whilst answering 68% of queries and is therefore an effective academic benchmark as well as a practical solution framework on which to build Bengali institutional support systems.

5.2 Future Work

In light of this, future work should replicate our study in some complementary but different ways. This first implies a strong motivation for a two-stage retrieval system in which the existing dense bi-encoder is followed by either cross-encoder reranker or hybrid denseLexical module to make preferred decisions over near-paraphrase collisions, generic-versus-specific mismatches, and lexical confusions. Second, the ability of retrieval formulation itself is expanded by dual indexing of questions, as well as an explicit use of topic metadata and hard-negative mining in order to discriminate more effectively between institution intents that are very similar. Thirdly, while the BRAC University corpus provides a useful and methodologically sound with-institution benchmark, it is still only one institution we should encourage future work to assess cross-domain generalization across multiple Bengali educational or governmental or public-service FAQ repositories. Fourth, further work on calibration and selective prediction is required to ensure that abstention policies are more reliable in high-stakes institutional settings where an incorrect response will cost societally more than seconds lapsed unasked. Fifth, robustness analysis should extend from pure-character deletion or transposition to more realistic Bengali user-noise patterns such as phonetic transliteration, mixed-script Spelling, abbreviated forms, OCR artifacts and keyboard-layout-related noise. Last but not least, the most robust long-term direction is the embedding of retriever into retrieval-grounded conversational support system capable of returning top-k FAQ candidates, justifying its responses, abstaining given low confidence and delegating open cases to human operators migrating from benchmarked retrieval performance to actual Bengali-student-support deployment.

Chapter 6

Conclusion

We stated the problem of Bengali institutional FAQ answering as a dense bi-encoder retrieval pre-training task, and proposed an educational realistic benchmark derived from BRAC University FAQs with 10,264 raw records and 7,233 cleaned question–answer pairs in over 148 topic labels. We create a unified fine-tuning protocol on this benchmark to evaluate 18 Bengali-specific, multilingual, contrastive, and MLM-based sentence encoders. Our results highlight a stark consistent trend: contemporary multilingual contrastive retrievers surpass all baseline approaches for this task, both Bengali-specific MLM baselines and generic Max-Multilingual MLM encoders. Multilingual-e5-Large did in fact perform the best overall for all three retrieval metrics, yielding Acc@1 = 0.923 and MRR@10 = 0.949, but contrastively trained models as a set outperformed MMLM-only models by +0.175 Acc@1 ($p < 0.05$), which suggests that retrieval-oriented pre-training is more consequential to Bengali FAQ search than monolingual language specificity alone for these popular contrasts considered here.

In addition to retrieval accuracy at a top-line level, the study shows that real-world FAQ system design is not limited to evaluating surface-level metrics. The best models had not only more accurate ranking of the correct answers, but also with better rank stability, even stronger embedding-space separation as well a robustness to character-level noise. Simultaneously, the results show that the most accurate model is not always the best-calibrated : multilingual-e5-large-instruct had the highest confidence calibration and on some levels focused more on this than quality while multilingual-e5-small showed to carry an appealing quality–efficiency tradeoff for latency-constrained deployment. They then developed a confidence-based abstention rule to reach 96.4% precision answering about two-thirds of the queries, over and above the baseline results case. These findings together imply that dense retrieval already constitutes a solid basis for Bengali institutional support systems even when take into account the tradeoff between accuracy and efficiency as well as confidence estimation.

The analysis also indicates that the task has not yet been solved. Collisions such as near-paraphrases, mismatches between generic versus specific intent and lexical-form confusions remain problematic even for the best model, suggesting that more fine-grained institutional distinctions are too faint to be discerned through single-stage dense retrieval. The corresponding ablation results further corroborate that fine-tuning is necessary for any model, 128 tokens is a decent enough instantiation of sequence budget over this dataset, larger effective batch sizes mostly help, and

mean pooling has clear advantages over using CLS pooling for plain MLM encoders. Taken together, these findings naturally lend themselves to future work on two-stage retrieval, hybrid dense–lexical systems, topic-aware indexing, harder negative construction, broader multi-institution Bengali benchmarks and more realistic robustness testing with well-mangled transliteration data or mixed-script input and typical OCR-like noise. This work thus provides a strong empirical baseline for Bengali FAQ retrieval, and demonstrates how a relevant multilingual retrieval models can be properly fine-tuned and evaluated to serve as the backbone of practical low-resource educational QA systems.

Bibliography

[1] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[2] M. Stanojević and K. Sima'an, "BEER: Better evaluation as ranking," in *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT)*, 2014, pp. 414–419.

[3] M. Popović, "chrF: Character n-gram F-score for automatic MT evaluation," in *Proceedings of the Tenth Workshop on Statistical Machine Translation (WMT)*, 2015, pp. 392–395.

[4] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017, pp. 1321–1330.

[5] M. Henderson et al., "Efficient natural language response suggestion for Smart Reply," *arXiv preprint arXiv:1705.00652*, 2017.

[6] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 3982–3992.

[8] A. Conneau et al., "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 8440–8451.

[9] D. Kakwani et al., "IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 4948–4961.

[10] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[11] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4512–4525.

[12] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2021.

[13] Y. Qu et al., "RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021, pp. 5835–5847.

[14] A. Bhattacharjee et al., "BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla," in *Findings of the Association for Computational Linguistics: NAACL 2022*, Seattle, United States: Association for Computational Linguistics, 2022, pp. 1318–1327.

[15] S. M. S. Ekram et al., "Banglarqa: A benchmark dataset for under-resourced bangla language reading comprehension-based question answering with diverse question-answer types," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022, pp. 2518–2532.

[16] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic BERT sentence embedding," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, arXiv:2007.01852, 2022.

[17] J. Su, M. Zhu, A. Murtadha, S. Pan, B. Wen, and Y. Liu, "CoSENT: A more efficient sentence-pair approach," *arXiv preprint arXiv:2208.04589*, 2022.

[18] A. Bhattacharjee, T. Hasan, W. Ahmad, and R. Shahriyar, "Banglanlg and banglat5: Benchmarks and resources for evaluating low-resource natural language generation in bangla," in *Findings of the Association for Computational Linguistics: EACL 2023*, 2023, pp. 726–735.

[19] S. Doddapaneni et al., "Towards leaving no indic language behind: Building monolingual corpora, benchmark and models for indic languages," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 12 402–12 426.

[20] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, "Towards general text embeddings with multi-stage contrastive learning," *arXiv preprint arXiv:2308.03281*, 2023.

[21] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023, pp. 2014–2037.

[22] M. S. Shahriar, A. A. F. Chowdhury, M. A. Ehsan, and A. R. Kamal, "Question answer generation in bengali: Mitigating the scarcity of qa datasets in a low-resource language," in *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 430–441.

[23] H. Su et al., "One embedder, any task: Instruction-finetuned text embeddings," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 1102–1121.

[24] K. Tian et al., "Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[25] X. Zhang et al., "Miracl: A multilingual retrieval dataset covering 18 diverse languages," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1114–1131, 2023.

[26] A. Acharya, R. Murthy, V. Kumar, and J. Sen, "Hindi-beir: A large scale retrieval benchmark in hindi," *arXiv preprint arXiv:2408.09437*, 2024.

[27] P. BehnamGhader, V. Adlakha, M. Mosbach, D. Bahdanau, N. Chapados, and S. Reddy, "Llm2vec: Large language models are secretly powerful text encoders," *arXiv preprint arXiv:2404.05961*, 2024.

[28] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "BGE M3-Embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *arXiv preprint arXiv:2402.03216*, 2024.

[29] J. Do, J. Lee, and S.-w. Hwang, "Contrastivemix: Overcoming code-mixing dilemma in cross-lingual transfer for information retrieval," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, 2024, pp. 197–204.

[30] C.-W. Huang, C.-A. Li, T.-Y. Hsu, C.-Y. Hsu, and Y.-N. Chen, "Unsupervised multilingual dense retrieval via generative pseudo labeling," in *Findings of the Association for Computational Linguistics: EACL 2024*, 2024, pp. 736–746.

[31] R. Jha et al., "Jina-colbert-v2: A general-purpose multilingual late interaction retriever," in *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, 2024, pp. 159–166.

[32] M. R. Kabir, M. M. R. Nabil, and M. A. Khan, "Banglaembed: Efficient sentence embedding models for a low-resource language using cross-lingual distillation techniques," in *2024 7th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI)*, IEEE, 2024, pp. 1–6.

[33] J. Lee, Y. Jung, and S.-w. Hwang, "Commit: Code-mixing english-centric large language model for multilingual instruction tuning," in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 3130–3137.

[34] M.-L. M.-F. Multi-Granularity, "M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *arXiv preprint arXiv:2402.03216*, 2024.

[35] N. Thakur, J. Ni, G. H. Ábrego, J. Wieting, J. Lin, and D. Cer, "Leveraging llms for synthesizing training data across many languages in multilingual dense retrieval," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 7699–7724.

[36] M. S. Uddin et al., *Bangla-sentence-transformer*, Hugging Face Model Card, 2024. [Online]. Available: https://huggingface.co/shihab17/bangla-sentence-transformer.

[37] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Improving text embeddings with large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 11 897–11 916.

[38] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Multilingual E5 text embeddings: A technical report," *arXiv preprint arXiv:2402.05672*, 2024.

[39] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Multilingual e5 text embeddings: A technical report," *arXiv preprint arXiv:2402.05672*, 2024.

[40] X. Zhang et al., "Mgte: Generalized long-context text representation and reranking models for multilingual text retrieval," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2024, pp. 1393–1412.

[41] T. Aarsen, *Train 400x faster static embedding models with Sentence Transformers*, Hugging Face Blog, 2025. [Online]. Available: https://huggingface.co/blog/static-embeddings.

[42] A. Acharya, R. Murthy, V. Kumar, and J. Sen, "Benchmarking and building zero-shot hindi retrieval model with hindi-beir and nllb-e5," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025, pp. 4328–4348.

[43] A. Ahmad, L. Adiba, N. Rasul, M. T. R. Laskar, and S. Ahmed, "Bncontextqa: Benchmarking long-context question answering and challenges in bangla," in *Proceedings of the Second Workshop on Bangla Language Processing (BLP-2025)*, 2025, pp. 357–365.

[44] M. Dinzinger, L. Caspari, K. Ghosh Dastidar, J. Mitrović, and M. Granitzer, "Webfaq: A multilingual collection of natural q&a datasets for dense retrieval," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025, pp. 3802–3811.

[45] K. Enevoldsen et al., "Mmteb: Massive multilingual text embedding benchmark," *arXiv preprint arXiv:2502.13595*, 2025.

[46] M. B. Jagadeeshan, P. Raj, and P. Goyal, "Anveshana: A new benchmark dataset for cross-lingual information retrieval on english queries and sanskrit documents," in *Computational Sanskrit and Digital Humanities-World Sanskrit Conference 2025*, 2025, pp. 161–180.

[47] J. Lee et al., "Gemini embedding: Generalizable embeddings from gemini," *arXiv preprint arXiv:2503.07891*, 2025.

[48] J. Trienes, A. Derzhanskaia, R. Schwarzkopf, M. Mühling, J. Schlötterer, and C. Seifert, "Marcel: A lightweight and open-source conversational agent for university student support," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2025, pp. 181–195.

[49] M. Dinzinger, L. Caspari, A. Salman, I. Topi, J. Mitrović, and M. Granitzer, "Webfaq 2.0: A multilingual qa dataset with mined hard negatives for dense retrieval," *arXiv preprint arXiv:2602.17327*, 2026.

[50] A. Eyasir, T. Ahmed, and M. Ibrahim, "Nctb-qa: A large-scale bangla educational question answering dataset and benchmarking performance," *arXiv preprint arXiv:2603.05462*, 2026.

Appendix A Implementation Code Snippets

Environment Setup and Reproducibility

```
import os, re, gc, json, time, random, warnings
from collections import Counter

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import torch

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA

from sentence_transformers import (
    SentenceTransformer, InputExample, losses, evaluation, models
)

from torch.utils.data import DataLoader
from transformers import AutoTokenizer

warnings.filterwarnings("ignore")

sns.set_style("whitegrid")
plt.rcParams["figure.dpi"] = 110
plt.rcParams["savefig.dpi"] = 140
plt.rcParams["figure.facecolor"] = "white"

SEED = 42
np.random.seed(SEED)
torch.manual_seed(SEED)
random.seed(SEED)

# Pin computation to a single GPU to avoid device mismatch.
if torch.cuda.is_available():
    os.environ["CUDA_VISIBLE_DEVICES"] = "0"
```

```
    torch.cuda.set_device(0)
    DEVICE = "cuda:0"
else:
    DEVICE = "cpu"

print(f"Device: {DEVICE}")

DATA_PATH = "/kaggle/input/datasets/maxdenis/brac-data/BRAC FAQ 10k Data.
    xlsx"

ROOT      = "/kaggle/working"
EDA_DIR   = f"{ROOT}/eda"
OUT_ROOT  = f"{ROOT}/brac_models"
LOG_ROOT  = f"{ROOT}/logs"
EVAL_DIR  = f"{ROOT}/evaluation"

for d in (EDA_DIR, OUT_ROOT, LOG_ROOT, EVAL_DIR):
    os.makedirs(d, exist_ok=True)

MAX_SEQ_LEN = 128
BATCH_SIZE  = 16
EPOCHS      = 3
LR          = 2e-5
```

Listing 6.1: Library imports, reproducibility settings, and GPU configuration.

Model Zoo Configuration

```
ENABLED_MODELS = None

MODELS = [
    {
        "name": "bangla-sbert",
        "hf_id": "shihab17/bangla-sentence-transformer",
        "kind": "st",
        "note": "Bengali-native sentence transformer"
    },
    {
        "name": "banglabert-buet",
        "hf_id": "csebuetnlp/banglabert",
        "kind": "bert",
        "note": "BUET BanglaBERT ELECTRA-base Bengali model"
    },
    {
        "name": "bangla-bert",
        "hf_id": "sagorsarker/bangla-bert-base",
        "kind": "bert",
        "note": "Bengali BERT-base model"
    },
    {
        "name": "minilm-multilingual",
```

```
        "hf_id": "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2",
        "kind": "st",
        "note": "Small multilingual MiniLM sentence transformer"
    },
    {
        "name": "mpnet-multilingual",
        "hf_id": "sentence-transformers/paraphrase-multilingual-mpnet-base-v2",
        "kind": "st",
        "note": "Multilingual MPNet sentence transformer"
    },
    {
        "name": "distiluse-multilingual",
        "hf_id": "sentence-transformers/distiluse-base-multilingual-cased-v2",
        "kind": "st",
        "note": "Distilled multilingual USE model"
    },
    {
        "name": "labse",
        "hf_id": "sentence-transformers/LaBSE",
        "kind": "st",
        "note": "Language-agnostic BERT sentence embedding model"
    },
    {
        "name": "e5-small",
        "hf_id": "intfloat/multilingual-e5-small",
        "kind": "st",
        "query_prefix": "query: ",
        "doc_prefix": "passage: ",
        "note": "Small multilingual E5 retriever"
    },
    {
        "name": "e5-base",
        "hf_id": "intfloat/multilingual-e5-base",
        "kind": "st",
        "query_prefix": "query: ",
        "doc_prefix": "passage: ",
        "note": "Base multilingual E5 retriever"
    },
    {
        "name": "xlm-roberta",
        "hf_id": "xlm-roberta-base",
        "kind": "bert",
        "note": "XLM-R base with mean pooling"
    },
    {
        "name": "indic-bert",
        "hf_id": "ai4bharat/indic-bert",
        "kind": "bert",
```

```
        "note": "Indic-BERT ALBERT-based multilingual Indic model"
    },
]

if ENABLED_MODELS is not None:
    MODELS = [m for m in MODELS if m["name"] in ENABLED_MODELS]

PREFIX_MAP = {
    c["name"]: (c.get("query_prefix", ""), c.get("doc_prefix", ""))
    for c in MODELS
}
```

Listing 6.2: Model zoo containing Bengali-specialized, multilingual, E5, and baseline retrievers.

Safe Utility Functions

```
def safe_read_csv(path):
    """
    Return a DataFrame if the file exists and is readable.
    Otherwise return None without crashing.
    """
    if not os.path.exists(path) or os.path.getsize(path) == 0:
        return None

    try:
        df_ = pd.read_csv(path)
        return df_ if len(df_) > 0 else None
    except (pd.errors.EmptyDataError, pd.errors.ParserError):
        return None


def free_mem():
    """
    Release Python and GPU memory after each model run.
    """
    gc.collect()

    if torch.cuda.is_available():
        torch.cuda.empty_cache()


def load_st_single_gpu(path_or_hf_id):
    """
    Load a SentenceTransformer model and force it to cuda:0.
    """
    model = SentenceTransformer(path_or_hf_id, device=DEVICE)
    model.to(DEVICE)
    model.max_seq_length = MAX_SEQ_LEN

```

```
    return model
```

Listing 6.3: Utility functions for safe CSV reading, memory cleanup, and single-GPU model loading.

Data Loading, Cleaning, and Quality Analysis

```
raw = pd.read_excel(DATA_PATH)
raw.columns = [c.strip() for c in raw.columns]

print("Raw shape:", raw.shape)
print("Columns:", raw.columns.tolist())

quality = {
    "total_rows":       len(raw),
    "null_topic":       int(raw["topic"].isna().sum()),
    "null_question":    int(raw["question"].isna().sum()),
    "null_answer":      int(raw["answer"].isna().sum()),
    "exact_duplicates": int(raw.duplicated().sum()),
    "dup_questions":    int(raw["question"].duplicated(keep=False).sum()),
    "dup_answers":      int(raw["answer"].duplicated(keep=False).sum()),
}

print("Data-quality summary:")
for k, v in quality.items():
    print(f"{k:20s}: {v:,}")

df = raw.dropna(subset=["question", "answer"]).copy()

df["question"] = df["question"].astype(str).str.strip()
df["answer"]   = df["answer"].astype(str).str.strip()
df["topic"]    = df["topic"].fillna("__UNKNOWN__").astype(str).str.strip()

df = df.drop_duplicates(subset=["question", "answer"]).reset_index(drop=
    True)

df["q_chars"] = df["question"].str.len()
df["a_chars"] = df["answer"].str.len()
df["q_words"] = df["question"].str.split().str.len()
df["a_words"] = df["answer"].str.split().str.len()

print("Cleaned dataset shape:", df.shape)

df[["q_chars", "a_chars", "q_words", "a_words"]] \
    .describe() \
    .round(2) \
    .to_csv(f"{EDA_DIR}/length_stats.csv")
```

Listing 6.4: Dataset loading, column cleaning, duplicate removal, and basic quality statistics.

Script Composition, Vocabulary, and Tokenizer Budget

```
def script_mix(text):
    """
    Measure Bengali, English, and digit character ratios.
    """
    bn = sum(1 for c in text if 0x0980 <= ord(c) <= 0x09FF)
    en = sum(1 for c in text if c.isascii() and c.isalpha())
    dg = sum(1 for c in text if c.isdigit())

    total = max(1, len(text))

    return bn / total, en / total, dg / total


q_mix = pd.DataFrame(
    df["question"].apply(script_mix).tolist(),
    columns=["bn", "en", "dg"]
)

a_mix = pd.DataFrame(
    df["answer"].apply(script_mix).tolist(),
    columns=["bn", "en", "dg"]
)


def tokenize(text):
    """
    Simple tokenization for vocabulary analysis.
    """
    text = re.sub(r"[,.?!()\"'\-:;]", " ", text)
    return [t for t in text.split() if len(t) > 1]


q_tokens = [t for s in df["question"] for t in tokenize(s)]
a_tokens = [t for s in df["answer"] for t in tokenize(s)]

q_vocab = Counter(q_tokens)
a_vocab = Counter(a_tokens)

tokenizer = AutoTokenizer.from_pretrained(
    "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2"
)

q_tok_len = [
    len(tokenizer.encode(x))
    for x in df["question"].sample(3000, random_state=0)
]

a_tok_len = [
    len(tokenizer.encode(x))
```

```
    for x in df["answer"].sample(3000, random_state=0)
]

pct_trunc_q = float(np.mean(np.array(q_tok_len) > MAX_SEQ_LEN) * 100)
pct_trunc_a = float(np.mean(np.array(a_tok_len) > MAX_SEQ_LEN) * 100)

print(f"Truncation at {MAX_SEQ_LEN}: Q={pct_trunc_q:.2f}% A={pct_trunc_a
    :.2f}%")

eda_summary = {
    "raw_rows": int(quality["total_rows"]),
    "clean_rows": int(len(df)),
    "unique_topics": int(df["topic"].nunique()),
    "avg_q_words": float(df["q_words"].mean()),
    "avg_a_words": float(df["a_words"].mean()),
    "q_vocab_size": len(q_vocab),
    "a_vocab_size": len(a_vocab),
    "pct_q_truncated_128": pct_trunc_q,
    "pct_a_truncated_128": pct_trunc_a,
}

with open(f"{EDA_DIR}/eda_summary.json", "w") as f:
    json.dump(eda_summary, f, indent=2)

df.drop(columns=["q_chars", "a_chars", "q_words", "a_words"]) \
    .to_parquet(f"{EDA_DIR}/cleaned_dataset.parquet", index=False)
```

Listing 6.5: Script composition, vocabulary extraction, and tokenizer-length analysis.

Train–Validation Split

```
train_df, val_df = train_test_split(
    df.drop(columns=["q_chars", "a_chars", "q_words", "a_words"]),
    test_size=0.05,
    random_state=SEED
)

train_df = train_df.reset_index(drop=True)
val_df   = val_df.reset_index(drop=True)

train_df.to_parquet(f"{LOG_ROOT}/train_split.parquet", index=False)
val_df.to_parquet(f"{LOG_ROOT}/val_split.parquet", index=False)

print(f"Train: {len(train_df)}")
print(f"Validation: {len(val_df)}")
```

Listing 6.6: Creation of train and validation splits for retrieval fine-tuning.

Model Construction

```python
def build_model(cfg):
    """
    Build either a ready-made SentenceTransformer model or a transformer
    encoder with a mean-pooling sentence embedding head.
    """
    if cfg["kind"] == "st":
        model = SentenceTransformer(cfg["hf_id"], device=DEVICE)

    else:
        word_embedding = models.Transformer(
            cfg["hf_id"],
            max_seq_length=MAX_SEQ_LEN
        )

        pooling = models.Pooling(
            word_embedding.get_word_embedding_dimension(),
            pooling_mode_mean_tokens=True
        )

        model = SentenceTransformer(
            modules=[word_embedding, pooling],
            device=DEVICE
        )

    model.to(DEVICE)
    model.max_seq_length = MAX_SEQ_LEN

    return model
```

Listing 6.7: Model construction for sentence-transformer models and transformer backbones with mean pooling.

Training Pair Construction and Retrieval Evaluator

```python
def make_examples(df_, cfg):
    """
    Convert each question-answer pair into an InputExample.
    E5 models receive query and passage prefixes.
    """
    query_prefix = cfg.get("query_prefix", "")
    doc_prefix   = cfg.get("doc_prefix", "")

    examples = [
        InputExample(
            texts=[
                f"{query_prefix}{row['question']}",
                f"{doc_prefix}{row['answer']}"
            ]
        )
        for _, row in df_.iterrows()
    ]
```

```
18
19      return examples
20
21
22  def make_evaluator(val_df_, cfg, name):
23      """
24      Build an InformationRetrievalEvaluator where each question has
25      exactly one relevant answer.
26      """
27      query_prefix = cfg.get("query_prefix", "")
28      doc_prefix   = cfg.get("doc_prefix", "")
29
30      queries = {
31          f"q{i}": f"{query_prefix}{q}"
32          for i, q in enumerate(val_df_["question"].tolist())
33      }
34
35      corpus = {
36          f"a{i}": f"{doc_prefix}{a}"
37          for i, a in enumerate(val_df_["answer"].tolist())
38      }
39
40      relevant_docs = {
41          f"q{i}": {f"a{i}"}
42          for i in range(len(val_df_))
43      }
44
45      evaluator = evaluation.InformationRetrievalEvaluator(
46          queries=queries,
47          corpus=corpus,
48          relevant_docs=relevant_docs,
49          name=name,
50          show_progress_bar=False,
51          batch_size=32,
52          accuracy_at_k=[1, 3, 5, 10, 20],
53          precision_recall_at_k=[1, 3, 5, 10, 20],
54          mrr_at_k=[1, 5, 10],
55          ndcg_at_k=[3, 5, 10, 20],
56          map_at_k=[5, 10, 20]
57      )
58
59      return evaluator
```

Listing 6.8: Question-answer pair creation and information-retrieval evaluator construction.

Multi-Model Fine-Tuning

```
1  all_results = []
2
3  for cfg in MODELS:
```

```
    name = cfg["name"]

    print("\n" + "=" * 80)
    print(f"TRAIN: {name} ({cfg['hf_id']})")
    print("=" * 80)

    out_dir = os.path.join(OUT_ROOT, name)
    log_dir = os.path.join(LOG_ROOT, name)

    os.makedirs(log_dir, exist_ok=True)

    try:
        model = build_model(cfg)

        train_loader = DataLoader(
            make_examples(train_df, cfg),
            shuffle=True,
            batch_size=BATCH_SIZE
        )

        loss_fn = losses.MultipleNegativesRankingLoss(model)
        evaluator = make_evaluator(val_df, cfg, f"{name}-val")

        loss_log = []
        original_forward = loss_fn.forward
        counter = {"n": 0}

        def logged_forward(*args, **kwargs):
            output = original_forward(*args, **kwargs)
            counter["n"] += 1

            if counter["n"] % 10 == 0:
                try:
                    loss_log.append({
                        "step": counter["n"],
                        "loss": float(output.detach().cpu())
                    })
                except Exception:
                    pass

            return output

        loss_fn.forward = logged_forward

        metric_log = []
        original_eval = evaluator.__call__

        def logged_eval(model_, output_path=None, epoch=-1, steps=-1, *a,
    **kw):
            score = original_eval(
                model_,
```

```
            output_path=output_path,
            epoch=epoch,
            steps=steps,
            *a,
            **kw
        )

        try:
            metrics = evaluator.compute_metrices(model_)
            row = {"epoch": epoch, "step": steps}

            for fn_name, fn_dict in metrics.items():
                for k, v in fn_dict.items():
                    if isinstance(v, dict):
                        for kk, vv in v.items():
                            row[f"{fn_name}_{k}@{kk}"] = float(vv)
                    else:
                        row[f"{fn_name}_{k}"] = float(v)

            metric_log.append(row)

        except Exception as ex:
            print(f"Metric-log warning: {ex}")

        return score

    evaluator.__call__ = logged_eval

    try:
        evaluator(model, epoch=-1, steps=0)
    except Exception as ex:
        print(f"Baseline evaluation warning: {ex}")

    start_time = time.time()
    warmup_steps = int(0.1 * len(train_loader) * EPOCHS)

    model.fit(
        train_objectives=[(train_loader, loss_fn)],
        evaluator=evaluator,
        epochs=EPOCHS,
        warmup_steps=warmup_steps,
        optimizer_params={"lr": LR},
        output_path=out_dir,
        show_progress_bar=True,
        save_best_model=True,
        use_amp=True,
        evaluation_steps=max(1, len(train_loader) // 4)
    )

    train_sec = time.time() - start_time

```

```
        model = load_st_single_gpu(out_dir)

        final_metrics = evaluator.compute_metrices(model)
        cosine_metrics = final_metrics.get(
            "cos_sim",
            next(iter(final_metrics.values()))
        )

        def get_metric(d, key, k):
            try:
                return float(d[key][k])
            except (KeyError, TypeError):
                return None

        record = {
            "model": name,
            "hf_id": cfg["hf_id"],
            "train_sec": round(train_sec, 1),

            "accuracy@1":  get_metric(cosine_metrics, "accuracy@k", 1),
            "accuracy@3":  get_metric(cosine_metrics, "accuracy@k", 3),
            "accuracy@5":  get_metric(cosine_metrics, "accuracy@k", 5),
            "accuracy@10": get_metric(cosine_metrics, "accuracy@k", 10),
            "accuracy@20": get_metric(cosine_metrics, "accuracy@k", 20),

            "precision@1":  get_metric(cosine_metrics, "precision@k", 1),
            "precision@3":  get_metric(cosine_metrics, "precision@k", 3),
            "precision@5":  get_metric(cosine_metrics, "precision@k", 5),
            "precision@10": get_metric(cosine_metrics, "precision@k", 10),

            "recall@1":  get_metric(cosine_metrics, "recall@k", 1),
            "recall@3":  get_metric(cosine_metrics, "recall@k", 3),
            "recall@5":  get_metric(cosine_metrics, "recall@k", 5),
            "recall@10": get_metric(cosine_metrics, "recall@k", 10),
            "recall@20": get_metric(cosine_metrics, "recall@k", 20),

            "mrr@1":  get_metric(cosine_metrics, "mrr@k", 1),
            "mrr@5":  get_metric(cosine_metrics, "mrr@k", 5),
            "mrr@10": get_metric(cosine_metrics, "mrr@k", 10),

            "ndcg@3":  get_metric(cosine_metrics, "ndcg@k", 3),
            "ndcg@5":  get_metric(cosine_metrics, "ndcg@k", 5),
            "ndcg@10": get_metric(cosine_metrics, "ndcg@k", 10),
            "ndcg@20": get_metric(cosine_metrics, "ndcg@k", 20),

            "map@5":  get_metric(cosine_metrics, "map@k", 5),
            "map@10": get_metric(cosine_metrics, "map@k", 10),
            "map@20": get_metric(cosine_metrics, "map@k", 20),

            "params_M": round(
                sum(p.numel() for p in model.parameters()) / 1e6,
```

```
                1
            ),
            "embed_dim": model.get_sentence_embedding_dimension()
        }

        all_results.append(record)

        if loss_log:
            pd.DataFrame(loss_log).to_csv(
                f"{log_dir}/loss_curve.csv",
                index=False
            )

        if metric_log:
            pd.DataFrame(metric_log).to_csv(
                f"{log_dir}/metric_curve.csv",
                index=False
            )

        print(
            f"{name}: acc@1={record['accuracy@1']:.3f}, "
            f"mrr@10={record['mrr@10']:.3f}, "
            f"time={record['train_sec']}s"
        )

        del model
        free_mem()

    except Exception as error:
        print(f"{name} failed: {type(error).__name__}: {error}")

        all_results.append({
            "model": name,
            "hf_id": cfg["hf_id"],
            "error": str(error)
        })

        free_mem()
        continue


summary = pd.DataFrame(all_results)

if "accuracy@1" in summary.columns:
    summary = summary.sort_values(
        "accuracy@1",
        ascending=False,
        na_position="last"
    )

summary.to_csv(f"{LOG_ROOT}/model_comparison.csv", index=False)
```

```
print(summary.to_string(index=False))
```

Listing 6.9: Fine-tuning all candidate models using Multiple Negatives Ranking Loss.

Per-Query Retrieval Prediction

```
def get_prefixes(name):
    return PREFIX_MAP.get(name, ("", ""))


BATCH = 64

val_questions = val_df["question"].tolist()
val_answers   = val_df["answer"].tolist()
val_topics    = val_df["topic"].tolist()

per_query = {}

for model_name in summary["model"]:
    query_prefix, doc_prefix = get_prefixes(model_name)

    model = SentenceTransformer(
        f"{OUT_ROOT}/{model_name}",
        device=DEVICE
    )

    question_embeddings = model.encode(
        [query_prefix + q for q in val_questions],
        batch_size=BATCH,
        convert_to_numpy=True,
        normalize_embeddings=True
    )

    answer_embeddings = model.encode(
        [doc_prefix + a for a in val_answers],
        batch_size=BATCH,
        convert_to_numpy=True,
        normalize_embeddings=True
    )

    similarity = question_embeddings @ answer_embeddings.T

    gold_scores = similarity[
        np.arange(len(val_questions)),
        np.arange(len(val_questions))
    ]

    ranks = (similarity >= gold_scores[:, None]).sum(axis=1)

    top1_idx = similarity.argmax(axis=1)
```

```
    top1_score = similarity.max(axis=1)
    top2_score = np.sort(similarity, axis=1)[:, -2]

    per_query[model_name] = pd.DataFrame({
        "qid": np.arange(len(val_questions)),
        "question": val_questions,
        "gold_answer": val_answers,
        "topic": val_topics,
        "top1_idx": top1_idx,
        "predicted_answer": [val_answers[i] for i in top1_idx],
        "predicted_topic": [val_topics[i] for i in top1_idx],
        "top1_score": top1_score,
        "gold_score": gold_scores,
        "margin": top1_score - top2_score,
        "rank": ranks,
        "correct@1": ranks == 1
    })

    per_query[model_name].to_csv(
        f"{EVAL_DIR}/per_query_{model_name}.csv",
        index=False
    )
```

Listing 6.10: Computing similarity matrix, gold rank, prediction, margin, and correctness for each query.

Extended Evaluation Metrics

```
try:
    import sacrebleu
except ImportError:
    os.system("pip install -q sacrebleu")
    import sacrebleu

from sklearn.metrics import brier_score_loss


def chrf_score(prediction, reference):
    """
    chrF similarity between predicted answer and gold answer.
    """
    try:
        return sacrebleu.sentence_chrf(
            prediction,
            [reference]
        ).score / 100.0
    except Exception:
        return 0.0


def ece_score(confidences, correct, n_bins=15):
```

```
    """
    Expected Calibration Error.
    """
    confidences = np.asarray(confidences)
    correct = np.asarray(correct, dtype=float)

    bins = np.linspace(0, 1, n_bins + 1)
    ece = 0.0
    n = len(confidences)

    for lower, upper in zip(bins[:-1], bins[1:]):
        mask = (confidences > lower) & (confidences <= upper)

        if mask.sum() == 0:
            continue

        bin_accuracy = correct[mask].mean()
        bin_confidence = confidences[mask].mean()

        ece += (mask.sum() / n) * abs(bin_accuracy - bin_confidence)

    return float(ece)


extra_metrics = []

for model_name in summary["model"]:
    df_model = per_query[model_name]

    ranks = df_model["rank"].to_numpy()
    correct = (ranks == 1).astype(int)
    scores = df_model["top1_score"].to_numpy()

    hit1  = float((ranks <= 1).mean())
    hit3  = float((ranks <= 3).mean())
    hit5  = float((ranks <= 5).mean())
    hit10 = float((ranks <= 10).mean())

    failure_rate_top10 = float((ranks > 10).mean())

    score_min = scores.min()
    score_max = scores.max()

    scores_01 = (scores - score_min) / (score_max - score_min + 1e-9)

    ece = ece_score(scores_01, correct)
    brier = float(brier_score_loss(correct, scores_01))

    text_similarities = [
        chrf_score(pred, gold)
        for pred, gold in zip(
```

```
            df_model["predicted_answer"],
            df_model["gold_answer"]
        )
    ]

    extra_metrics.append({
        "model": model_name,
        "hit@1": round(hit1, 4),
        "hit@3": round(hit3, 4),
        "hit@5": round(hit5, 4),
        "hit@10": round(hit10, 4),
        "failure_rate_top10": round(failure_rate_top10, 4),
        "ECE": round(ece, 4),
        "Brier": round(brier, 4),
        "rank_mean": round(float(ranks.mean()), 2),
        "rank_median": float(np.median(ranks)),
        "rank_p90": float(np.percentile(ranks, 90)),
        "chrf_mean": round(float(np.mean(text_similarities)), 4)
    })


extra_df = pd.DataFrame(extra_metrics)

summary_full = summary.merge(
    extra_df,
    on="model",
    how="left"
)

summary_full.to_csv(
    f"{EVAL_DIR}/summary_full_metrics.csv",
    index=False
)

print(summary_full.to_string(index=False))
```

Listing 6.11: Additional retrieval, calibration, and ranking metrics for detailed model evaluation.

Robustness Testing Under Character-Level Noise

```
def corrupt(text, rate=0.1):
    """
    Apply character-level noise by randomly dropping or swapping
    characters.
    """
    chars = list(text)

    if len(chars) == 0:
        return text
```

```
9
10      n_edits = max(1, int(len(chars) * rate))
11
12      for _ in range(n_edits):
13          if len(chars) == 0:
14              break
15
16          i = random.randrange(len(chars))
17          operation = random.choice(["drop", "swap"])
18
19          if operation == "drop" and len(chars) > 1:
20              chars.pop(i)
21
22          elif operation == "swap" and i < len(chars) - 1:
23              chars[i], chars[i + 1] = chars[i + 1], chars[i]
24
25      return "".join(chars)
26
27
28  noise_rates = [0.00, 0.05, 0.10, 0.15, 0.20, 0.30]
29
30  robustness = {}
31
32  for model_name in summary["model"]:
33      query_prefix, doc_prefix = get_prefixes(model_name)
34
35      model = SentenceTransformer(
36          f"{OUT_ROOT}/{model_name}",
37          device=DEVICE
38      )
39
40      answer_embeddings = model.encode(
41          [doc_prefix + a for a in val_answers],
42          batch_size=BATCH,
43          convert_to_numpy=True,
44          normalize_embeddings=True
45      )
46
47      accuracies = []
48
49      for noise_rate in noise_rates:
50          noisy_questions = [
51              corrupt(q, noise_rate)
52              for q in val_questions
53          ]
54
55          question_embeddings = model.encode(
56              [query_prefix + q for q in noisy_questions],
57              batch_size=BATCH,
58              convert_to_numpy=True,
59              normalize_embeddings=True
```

```
        )

        similarity = question_embeddings @ answer_embeddings.T

        accuracy = (
            similarity.argmax(axis=1) == np.arange(len(val_questions))
        ).mean()

        accuracies.append(float(accuracy))

    robustness[model_name] = accuracies

robustness_df = pd.DataFrame(
    robustness,
    index=[f"noise_{r:.2f}" for r in noise_rates]
).T

robustness_df.to_csv(f"{EVAL_DIR}/robustness.csv")
print(robustness_df)
```

Listing 6.12: Robustness evaluation using character deletion and adjacent-character swap noise.

Inference Latency Benchmarking

```
latency = []

for model_name in summary["model"]:
    query_prefix, _ = get_prefixes(model_name)

    model = SentenceTransformer(
        f"{OUT_ROOT}/{model_name}",
        device=DEVICE
    )

    _ = model.encode(
        [query_prefix + "warmup"],
        convert_to_numpy=True
    )

    start = time.time()

    for _ in range(50):
        _ = model.encode(
            [query_prefix + val_questions[0]],
            convert_to_numpy=True
        )

    single_query_ms = (time.time() - start) / 50 * 1000

```

```
    start = time.time()

    _ = model.encode(
        [query_prefix + q for q in val_questions[:256]],
        batch_size=64,
        convert_to_numpy=True
    )

    batched_ms_per_item = (time.time() - start) * 1000 / 256

    latency.append({
        "model": model_name,
        "single_query_ms": round(single_query_ms, 2),
        "batched_ms_per_item": round(batched_ms_per_item, 2)
    })

latency_df = pd.DataFrame(latency)
latency_df.to_csv(f"{EVAL_DIR}/latency.csv", index=False)

print(latency_df)
```

Listing 6.13: Single-query and batched inference latency benchmarking.

Failure Case Extraction

```
failure_cases = []

for model_name in summary["model"]:
    df_model = per_query[model_name]

    errors = df_model[df_model["correct@1"] == False].copy()

    errors = errors.sort_values(
        ["rank", "margin"],
        ascending=[False, True]
    )

    errors["model"] = model_name

    failure_cases.append(
        errors[
            [
                "model",
                "qid",
                "topic",
                "question",
                "gold_answer",
                "predicted_answer",
                "predicted_topic",
                "rank",
```

```
                "top1_score",
                "gold_score",
                "margin"
            ]
        ].head(25)
    )

failure_df = pd.concat(failure_cases, ignore_index=True)

failure_df.to_csv(
    f"{EVAL_DIR}/failure_cases.csv",
    index=False
)

print(failure_df.head(20).to_string(index=False))
```

Listing 6.14: Extraction of representative retrieval failure cases for qualitative analysis.

FAQ Index Construction

```
winner = summary.iloc[0]["model"]

query_prefix, doc_prefix = get_prefixes(winner)

best_model = SentenceTransformer(
    f"{OUT_ROOT}/{winner}",
    device=DEVICE
)

faq_questions = df["question"].tolist()
faq_answers   = df["answer"].tolist()
faq_topics    = df["topic"].tolist()

faq_embeddings = best_model.encode(
    [doc_prefix + a for a in faq_answers],
    batch_size=64,
    convert_to_numpy=True,
    normalize_embeddings=True
)

print(f"Winner model: {winner}")
print(f"FAQ index size: {len(faq_embeddings)}")
```

Listing 6.15: Building the final FAQ embedding index using the best-performing model.

FAQ Question Answering API

```
def ask(user_question: str, top_k: int = 3, min_score: float = 0.55):
    """
    Retrieve the most relevant FAQ answer for a user question.
    If the similarity score is below the threshold, return a fallback
    answer.
    """
    question_embedding = best_model.encode(
        [query_prefix + user_question],
        convert_to_numpy=True,
        normalize_embeddings=True
    )

    scores = (faq_embeddings @ question_embedding.T).ravel()

    top_indices = np.argpartition(-scores, top_k)[:top_k]
    top_indices = top_indices[np.argsort(-scores[top_indices])]

    best_index = int(top_indices[0])
    best_score = float(scores[best_index])

    if best_score >= min_score:
        answer = faq_answers[best_index]
    else:
        answer = "Sorry, a reliable answer could not be found."

    candidates = [
        {
            "score": float(scores[i]),
            "topic": faq_topics[i],
            "question": faq_questions[i],
            "answer": faq_answers[i]
        }
        for i in top_indices
    ]

    return {
        "answer": answer,
        "best_score": best_score,
        "candidates": candidates
    }


example_result = ask(
    "How can I get admission information?",
    top_k=3,
    min_score=0.55
)

print(example_result["answer"])
```

```
print(example_result["best_score"])
```

Listing 6.16: The ask function retrieves the most relevant FAQ answer for a user question.