A Cross-Lingual Information Retrieval System for Bangla–English News Corpora

Aimaan Ahmed - 210041204*, Afra Anika - 210041206*
Nabila Newaz - 210041216*, Tanjil Hasan Khan - 210041246*
*Department of Computer Science and Engineering
Islamic University of Technology
Gazipur, Dhaka
{aimaanahmed, afraanika21, nabilanewaz, tanjilhasan}@iut-dhaka.edu

***Abstract*—Cross-Lingual Information Retrieval (CLIR) is a method enables users to retrieve relevant documents written in a language different from their query language. In multilingual environments such as Bangladesh, users frequently search using Bangla, English, or mixed-language queries ”Banglish” . Monolingual search systems fail to address such cross-language and semantic mismatches.**

In this work, we develop a Bangla–English CLIR system over a curated multilingual news corpus containing over 9550 documents (3855 from English and 5695 from Bangla) . Our system integrates lexical , fuzzy matching, and multilingual embedding-based semantic retrieval models. We further implement query translation, named entity mapping, hybrid ranking, and confidence-aware scoring. The system is evaluated using Precision@10, Recall@50, nDCG@10, and Mean Reciprocal Rank (MRR), along with detailed error analysis of translation drift, named entity mismatches, cross-script ambiguity, and code-switching scenarios.

Our findings demonstrate that embedding-based retrieval significantly improves cross-lingual performance, while hybrid ranking offers balanced robustness.The implementation is publicly available [14].

***Index Terms*—Cross-Lingual Information Retrieval, Multilingual Search, BM25, Semantic Retrieval, Bangla NLP, Named Entity Mapping**

I. Introduction and Motivation

The rapid growth of digital content has led to large-scale multilingual information across news platforms and online media. In multilingual settings such as Bangladesh, users frequently issue queries in Bangla, English, or a mixture of both. However, most traditional information retrieval (IR) systems operate in monolingual environments, limiting access to relevant documents written in other languages.

Cross-Lingual Information Retrieval (CLIR) addresses this limitation by enabling users to submit queries in one language and retrieve documents written in another. Despite recent advances in multilingual NLP, practical CLIR systems face several challenges. These include translation drift, named entity mismatches across scripts, inconsistent transliteration, and code-switching within queries. Such issues reduce retrieval accuracy and complicate semantic alignment.

Recent multilingual embedding models (e.g., LaBSE and XLM-R) provide strong cross-lingual semantic representations. However, purely semantic approaches may sacrifice lexical precision, while classical lexical methods such as BM25 fail across languages. Therefore, an effective CLIR system requires a balanced integration of lexical, fuzzy, and semantic retrieval techniques.

In this work, we develop a Bangla–English CLIR system over a curated multilingual news corpus containing more than 9550 documents. Our system combines: (i) lexical retrieval using BM25, (ii) fuzzy and transliteration-based matching, and (iii) multilingual embedding-based semantic similarity. We further incorporate query translation, optional named entity mapping, hybrid ranking with normalized confidence scores, and a low-confidence warning mechanism.

The system is evaluated using standard IR metrics, including Precision@10, Recall@50, nDCG@10, and Mean Reciprocal Rank (MRR). Through empirical analysis, we examine retrieval accuracy, computational trade-offs, and practical challenges specific to Bangla–English cross-lingual retrieval. Our goal is to provide a reproducible and practical CLIR framework grounded in real-world multilingual news data.

II. Literature Review

A. Multilingual Retrieval-Augmented Generation for Culturally-Sensitive Tasks (ACL Findings 2025)

[1]

Authors: Bryan Li, Fiona Luo, Samar Haider, Adwait Agashe, Tammy Li, Runqi Liu, Muqing Miao, Shriya Ramakrishnan, Yuan Yuan, Chris Callison-Burch
Year: 2025

Main Technique: Multilingual retrieval-augmented generation using multilingual embeddings and retrieval pipelines.

This paper introduces the *BORDIRLINES* benchmark, a multilingual dataset covering 49 languages to evaluate the robustness of retrieval-augmented generation (RAG) systems in cross-lingual settings. This framework retrieves relevant documents from multilingual knowledge bases using embedding-based semantic retrieval and generates responses using LLMs. The study evaluates multiple retrieval strategies, including monolingual and multilingual retrieval, and demonstrates that multilingual retrieval improves factual precision, consistency, and reduces geopolitical bias. The results indicate that accessing information across languages enables models to gain diverse perspectives and improves overall reliability. In their methodology for *CLIR*, they used two multilingual embedding systems: OpenAI embeddings [7] and BGE-M3. [8] This paper helped us to select and evaluate embedding models for our methodology.

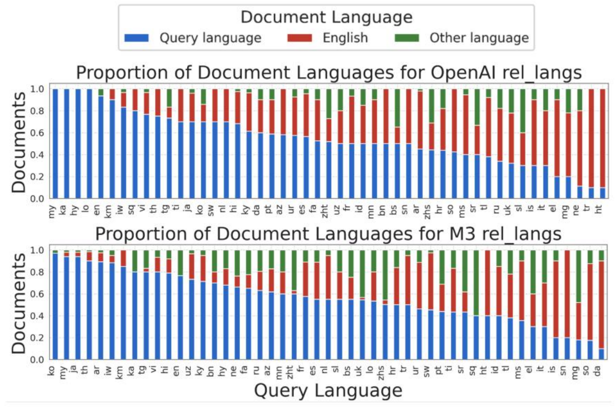


Figure 1: Proportion of document languages retrieved per query language, using OpenAI and M3 embeddings [1].

B. *Cross-lingual Retrieval Augmented In-context Learning for Bangla (Li et al., 2023)*

[2]

Authors: Xiaoqian Li, Ercong Nie, Sheng Liang
Year: 2023
Main Technique: Cross-lingual semantic retrieval using multilingual sentence embeddings and retrieval-augmented prompting.

This paper proposes a cross-lingual retrieval augmentation pipeline designed to enhance Bangla natural language processing performance. The system retrieves semantically similar examples from high-resource languages such as English using multilingual embedding models and add them into the input context to improve prediction accuracy. The retrieval process maps the Bangla input into a shared multilingual embedding space and retrieves relevant samples on the basis of cosine similarity. Experimental results demonstrate significant improvements in classification and summarization tasks for Bangla, highlighting the effectiveness of cross-lingual retrieval in low-resource language settings.

This paper helped us demonstrate the effectiveness of multilingual embedding-based retrieval for Bangla. It provides strong evidence that semantic retrieval techniques can overcome resource limitations and improve cross-lingual retrieval accuracy.

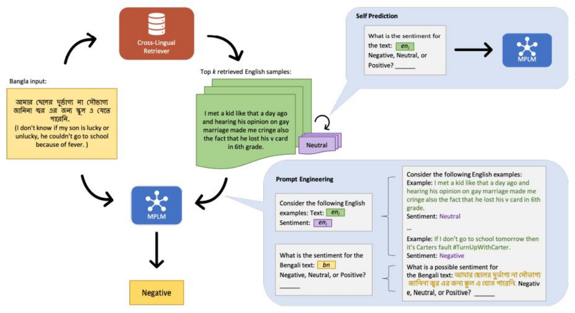

Cross-Lingual Retriever
Bangla input:
আমার ছেলের দুর্ভাগ্য না সৌভাগ্য জানিনা জ্বর এর জন্য স্কুল এ যেতে পারেনি.
(I don't know if my son is lucky or unlucky, he couldn't go to school because of fever.)
Top k retrieved English samples:
I met a kid like that a day ago and hearing his opinion on gay marriage made me cringe also the fact that he lost his v card in 6th grade.
Neutral
Self Prediction
What is the sentiment for the text: en_i Negative, Neutral, or Positive? ______
MPLM
MPLM
Negative
Prompt Engineering
Consider the following English examples: Text: en_i Sentiment: en_i
What is the sentiment for the Bengali text: bn Negative, Neutral, or Positive? ______
Consider the following English examples:
Example: I met a kid like that a day ago and hearing his opinion on gay marriage made me cringe also the fact that he lost his v card in 6th grade.
Sentiment: Neutral
...
Example: If I don't go to school tomorrow then it's Carters fault #TurnUpWithCarter.
Sentiment: Negative
What is a possible sentiment for the Bengali text: আমার ছেলের দুর্ভাগ্য না সৌভাগ্য জানিনা জ্বর এর জন্য স্কুল এ যেতে পারেনি. Negative, Neutral, or Positive? ______

Figure 2: Overview of the PARC pipeline for low-resource languages. An LRL query retrieves semantically similar HRL samples via a cross-lingual retriever; the retrieved sample and its label are added into a retrieval-augmented prompt for multilingual language model (MPLM [9], [10]) prediction.

C. *Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation (Reimers and Gurevych, 2020)*

[3]

Authors: Nils Reimers, Iryna Gurevych
Year: 2020
Main Technique: Multilingual knowledge distillation to align sentence embeddings across languages.

This paper introduces a knowledge distillation approach to extend monolingual sentence embedding models into multilingual embedding models. The method trains a student model using parallel translated sentences so that semantically identical sentences in different languages are mapped to the same location in the embedding space. This enables semantic similarity comparison across languages without requiring explicit translation. The proposed method significantly improves multilingual embedding performance across more than 50 languages.

This work is foundational for multilingual semantic retrieval without translation. The proposed Bangla–English CLIR system relies on multilingual embeddings to match queries and documents across languages. This paper helped us to establish the theoretical and technical basis for embedding-based cross-lingual retrieval.

D. *Language-agnostic BERT Sentence Embedding (LaBSE) (Feng et al., 2022)*

[4]

Authors: Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, Wei Wang
Year: 2022
Main Technique: Transformer-based multilingual sentence embedding using a dual encoder architecture.

This paper introduces the Language-agnostic BERT Sentence Embedding (LaBSE) model, which maps sentences from different languages into a shared semantic vector space. The model is trained on large-scale parallel translation data using transformer architectures. It enables accurate cross-lingual semantic similarity comparison and supports more than 100 languages. LaBSE achieves state-of-the-art performance in multilingual retrieval and semantic similarity tasks.

This paper is important because it demonstrates the effectiveness of multilingual embeddings for cross-lingual retrieval. The proposed Bangla–English CLIR system relies on similar embedding models to retrieve relevant documents across languages.

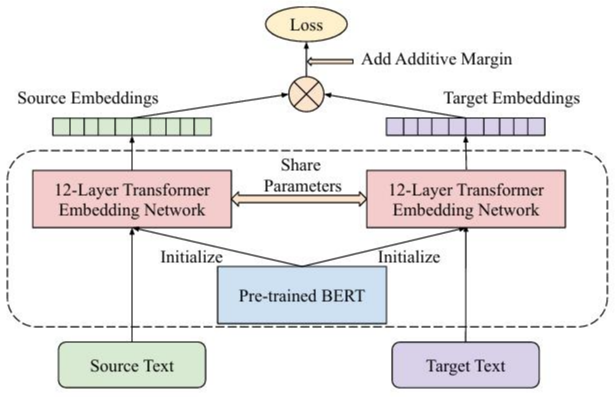


Figure 3: Dual encoder model with BERT based encoding modules [4].

E. *Dense Passage Retrieval for Open-Domain Question Answering (Karpukhin et al., 2020)*

[5]

Authors: Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Danqi Chen, Wen-tau Yih
Year: 2020
Main Technique: Dense vector retrieval using dual encoder transformer models.

This paper introduces Dense Passage Retrieval (DPR), which uses transformer-based dual encoder models to encode queries and documents into dense vector representations. Retrieval is performed using vector similarity instead of lexical matching. The method significantly improves retrieval accuracy compared to traditional lexical retrieval methods such as BM25. Dense retrieval enables semantic matching between queries and documents even when exact keywords do not match.

This work provides the foundation for modern semantic search systems. The proposed Bangla–English CLIR system uses embedding-based semantic retrieval, which follows the dense retrieval approach described in this paper.

F. *Cross-Lingual Information Retrieval: A Survey (Gupta et al., 2022)*

[6]

Authors: Sakshi Gupta, Ronak Pradeep, et al.
Year: 2022
Main Technique: Comprehensive survey of cross-lingual retrieval techniques.

This survey provides a comprehensive overview of cross-lingual information retrieval methods, including dictionary-based, translation-based, and embedding-based retrieval approaches. The paper highlights the advantages of neural embedding-based retrieval methods, which enable semantic matching across languages and improve retrieval accuracy. It also discusses evaluation methods and challenges in multilingual retrieval systems.

This paper provides important theoretical background and supports the use of embedding-based retrieval techniques. The proposed Bangla–English CLIR system follows modern embedding-based retrieval approaches discussed in this survey.

III. Methodology & Tools

A. *Dataset Construction and Preprocessing*

1) Multilingual News Corpus Construction: To build a realistic Bangla–English CLIR benchmark, we developed a scalable multilingual news crawling and normalization pipeline using the **Scrapy** framework. The corpus was collected from major Bangladeshi news portals covering politics, economy, sports, international news, and regional reporting.

Unlike translated or synthetic datasets, our corpus contains authentic journalistic content with heterogeneous HTML structures and natural language distribution. This makes it suitable for real-world cross-lingual retrieval evaluation [2], [6].

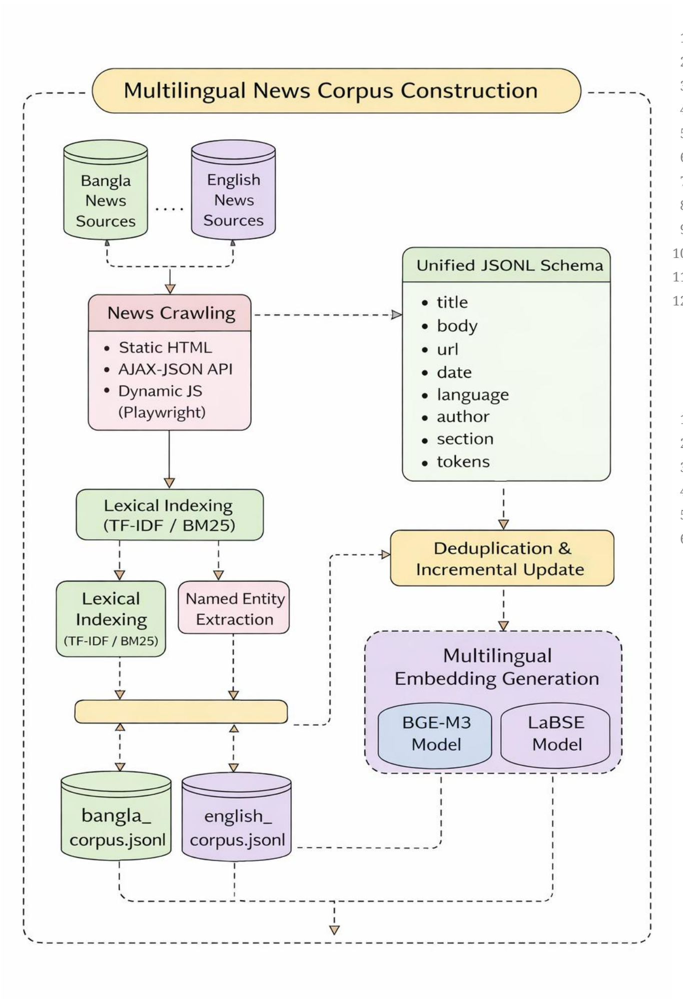


Figure 4: Dataset Construction and Preprocessing Pipeline

a) News Sources: The corpus integrates Bangla and English newspapers to ensure linguistic and topical diversity.

Bangla Sources: Bangla Tribune, Dhaka Post, Prothom Alo, Kaler Kantho, Daily Naya Diganta

English Sources: Daily Sun, New Age, Daily New Nation, Dhaka Tribune

Each source was crawled using category-specific spiders to maximize topical coverage while preserving source diversity.

b) Crawling Architecture: The crawling system supports heterogeneous website structures, including:

- Static HTML listing pages
- AJAX-driven pagination endpoints
- REST-style JSON APIs
- Dynamically rendered content (Playwright-enabled scraping)

For example, the Bangla Tribune crawler uses AJAX pagination to retrieve additional articles:

```
yield scrapy.Request(
    url=self.build_api_url(
        start=20,
        page_id=page_id,
        tags=tags
    ),
    callback=self.parse_api,
    headers={
        "X-Requested-With": "XMLHttpRequest",
        "Referer": response.url
    }
)
```

Prothom Alo articles are retrieved through offset-based pagination using their public JSON API:

```
params = {
    "item-type": "story",
    "offset": self.offset,
    "limit": 10
}
url = f"{API_BASE}?{urlencode(params)}"
```

For dynamically rendered websites such as Dhaka Post, we integrated Playwright with Scrapy to simulate user interaction and trigger “Load More” functionality.

This hybrid crawling strategy improves recall across websites with different rendering mechanisms.

c) Metadata Extraction and Schema Design: All scraped articles were normalized into a unified JSONL schema for cross-source consistency. Each document contains:

- `title`
- `body`
- `url`
- `date`
- `language`
- `author`
- `section`
- `tokens`

Token counts are computed at crawl time to support document length normalization for BM25 scoring [12]. Each output file follows a line-delimited JSON format (`.jsonl`), enabling efficient streaming, incremental indexing, and batch embedding generation.

d) Deduplication and Incremental Updates: To prevent duplicate entries across multiple crawling runs, a URL-based deduplication mechanism was implemented:

```
if url in existing_urls:
    skipped += 1
    continue
```

This enables incremental corpus expansion while maintaining data consistency.

Raw outputs were consolidated into two primary corpora:

- `bangla_corpus.jsonl`
- `english_corpus.jsonl`

e) Corpus Statistics: After normalization and cleaning, the final corpus contains:

- 5,695 Bangla documents
- 3,855 English documents

The multilingual and multi-domain nature of the dataset creates a realistic and challenging benchmark for CLIR evaluation [6].

f) Scrapy Project Architecture: The crawling system follows standard Scrapy design principles:

- **items.py:** Structured data containers for scraped articles.
- **middlewares.py:** Request–response interception hooks.
- **pipelines.py:** Item-level post-processing and validation.
- **settings.py:** Concurrency, throttling, and encoding configuration.
- **spiders/:** Domain-specific crawlers for each source.

Politeness and stability constraints were enforced through:

```
CONCURRENT_REQUESTS_PER_DOMAIN = 4
DOWNLOAD_DELAY = 2
ROBOTSTXT_OBEY = True
FEED_EXPORT_ENCODING = "utf-8"
```

This modular design allows scalable multilingual corpus expansion.

2) Lexical Index Construction: We construct classical inverted indices for both Bangla and English corpora to support TF-IDF and BM25 retrieval. Lexical retrieval remains a strong baseline in CLIR systems [6], [12].

a) Tokenization and Normalization: A lightweight language-aware tokenizer is implemented. English text is lowercased for case-insensitive matching, while Bangla casing is preserved. Punctuation is removed while retaining Bangla Unicode characters (U+0980–U+09FF).

Listing 1: Language-aware tokenization

```python
def tokenize(text, language):
  text = text.strip()

  if language == "english":
    text = text.lower()

  text = re.sub(r"[^\w\s\u0980-\u09FF]", " ", text)
  tokens = text.split()
  return tokens
```

b) Inverted Index Structure: The inverted index maps terms to document-frequency pairs:

$$\mathcal{I}(t) = \{(d_i, \mathrm{tf}_{t,d_i})\}$$

Document lengths are stored for BM25 normalization:

$$|d_i| = \text{number of tokens in document } d_i$$

Listing 2: Inverted index construction

```python
for term, freq in tf.items():
  inverted_index[term][doc_id] = freq
```

Table I: Lexical Index Statistics for Bangla and English Corpora

Language	Documents	Vocabulary Size	Avg. Doc Length
Bangla	5,695	80,610	353.70
English	3,855	41,444	414.01

3) Index Statistics:

a) Rationale: Lexical retrieval provides strong precision for exact term matching, especially for named entities and domain-specific terminology [6]. However, lexical methods alone cannot fully address cross-lingual mismatch. Combining lexical retrieval with dense embeddings improves recall and robustness [5].

4) Named Entity Extraction and Cross-Lingual Alignment: Named entities are critical in multilingual news retrieval, as script mismatches frequently cause lexical retrieval failures [6]. We extract named entities from both corpora and store them as structured metadata for query expansion and entity-aware ranking.

a) NER Models: We employ transformer-based multilingual NER models:

Listing 3: NER Pipeline Initialization

```python
from transformers import pipeline

bn_ner = pipeline(
  "ner",
  model="Davlan/xlm-roberta-base-wikiann-ner",
  aggregation_strategy="simple"
)

en_ner = pipeline(
  "ner",
  model="xlm-roberta-large-finetuned-conll03-
      english",
  aggregation_strategy="simple"
)
```

These models are based on multilingual transformer architectures that support cross-lingual representation learning [3].

b) Entity Grouping and Storage:

$$\mathcal{E}(d_i) = \{\text{PER} : [\ldots], \text{LOC} : [\ldots], \text{ORG} : [\ldots]\}$$

Listing 4: Entity Grouping

```python
def group_entities(entities):
  grouped = defaultdict(list)
  for ent in entities:
    label = ent["entity_group"]
```

```
        text = ent["word"]
        grouped[label].append(text)
    return dict(grouped)
```

Table II: Named Entity Coverage Statistics

Language	Total Documents	Docs with Entities
Bangla	5,695	5,665
English	3,855	3,843

c) Coverage Statistics: More than 99% of documents contain at least one named entity, highlighting the importance of entity-aware retrieval in news-domain CLIR systems.

Table III presents representative NER outputs from both languages. For formatting compatibility in IEEE two-column layout, Bangla entities are shown in transliterated form, while the underlying extraction was performed on original Unicode text.

Table III: Representative Named Entity Recognition Outputs

Language	Entity	Type	Confidence
Bangla	Sheikh Hasina	PER	0.9987
Bangla	Dhaka	LOC	0.9992
Bangla	Grameen Bank	ORG	0.9381
English	Joe Biden	PER	1.0000
English	New York City	LOC	1.0000
English	United Nations	ORG	1.0000

5) Multilingual Embedding Construction: Dense multilingual embeddings enable semantic cross-lingual retrieval by projecting multiple languages into a shared vector space [3], [5].

We generate document embeddings using BGE-M3 [8] and LaBSE [4].

a) Model Selection: BGE-M3 supports dense, sparse, and multi-vector retrieval paradigms [8]. LaBSE produces aligned multilingual sentence embeddings across 100+ languages [4].

b) Text Preparation: For BGE-M3, document representations are constructed by concatenating title and body:

$$\mathbf{d}_i = \texttt{title}_i \,||\, \mathsf{body}_i$$

For LaBSE, we encode the document body directly.

c) Batch Encoding and Storage: We used the following models to create the embeddings, they were evaluated in parallel to test their performance

Listing 5: bge-m3 embedding generation

```
model = BGEM3FlagModel("BAAI/bge-m3", use_fp16=True)

embeddings = model.encode(
    texts,
    batch_size=32,
    max_length=512
)["dense_vecs"]
```

```
np.save("bn_embeddings_bgem3.npy", embeddings)
```

Listing 6: LaBSE embedding generation

```
model = SentenceTransformer("sentence-transformers/
     LaBSE")

emb = model.encode(
    batch,
    normalize_embeddings=True
)
```

Embeddings are stored as NumPy arrays (.npy) with separate document ID mappings.

d) Retrieval Setup: Similarity is computed using cosine similarity:

$$\mathrm{sim}(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|}$$

Maintaining both BGE-M3 and LaBSE embeddings allows comparative evaluation of retrieval quality and alignment robustness.

B. Query Processing & Cross-Lingual Handling

1) Purpose: This module implements the query-side processing needed to support Bangla–English cross-lingual retrieval. Real queries may contain noisy formatting, transliterations, and named entities, which can cause cross-lingual mismatch [6]. The objective is to process a query written in either Bangla or English so that it can retrieve relevant documents regardless of document language. This design follows established CLIR practice and multilingual retrieval findings [1], [2], [6].

2) Pipeline: The query-processing pipeline follows the required steps below:

- Language Detection
- Normalization
- Query Conversion/Translation
- Query Expansion
- Named-Entity Mapping

1) **Language Detection** (Bangla vs. English). Bangla is detected via Unicode script range. This lightweight heuristic is reliable for Bangla–English queries because the scripts are disjoint, and it avoids heavy language-ID models. The detected label ℓ controls downstream actions such as lowercasing, translation direction, and rule-based expansion.

Listing 7: Language Detection using Unicode Range

```
def detect_language(text):
    for ch in text:
        if '\u0980' <= ch <= '\u09FF':
            return "bn"
    return "en"
```

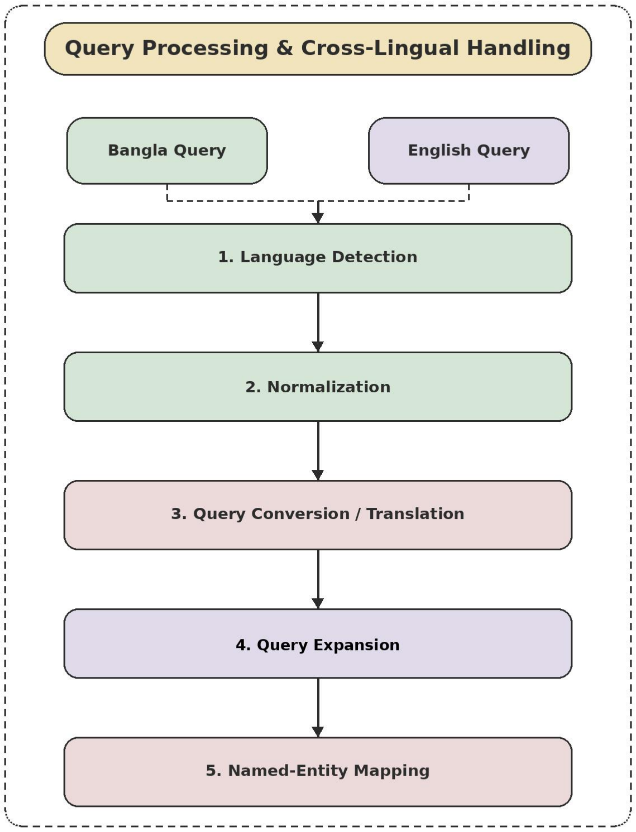


Figure 5: Query Processing & Cross-Lingual Handling.

Let the raw user query be denoted by q.

$$\ell = \mathcal{L}(q), \qquad \ell \in \{\mathrm{bn}, \mathrm{en}\}. \tag{1}$$

2) **Normalization** (lowercase + whitespace cleanup). English is lowercased; extra whitespace is removed. Stopword removal is optional [6]. Unicode NFC normalization is applied to reduce representation variance in Bangla characters and punctuation, improving stable matching across repeated queries. Normalization is kept conservative to avoid removing short but meaningful tokens (especially named entities).

Listing 8: Query Normalization

```python
import re, unicodedata

def normalize_query(q, lang):
   q = unicodedata.normalize("NFC", q).strip()
   if lang == "en":
      q = q.lower()
   q = re.sub(r"\s+", " ", q).strip()
   return q
```

$$q_{\mathrm{norm}} = \mathcal{N}(q, \ell), \tag{2}$$

where $\mathcal{N}(\cdot)$ applies whitespace normalization and English lowercasing (stopword removal is optional).

3) **Query Conversion / Translation**. A translated variant is generated to improve cross-lingual coverage; translation is wrapped with fallback handling [1], [6]. Because short queries can be ambiguous, the original query is retained alongside the converted variant to avoid error propagation from imperfect translation. This step directly increases recall for lexical matching when relevant documents exist primarily in the opposite language.

Listing 9: Robust Translation Wrapper

```python
from deep_translator import GoogleTranslator

def safe_translate(q, src, tgt):
   try:
      return GoogleTranslator(source=src, target
          =tgt).translate(q)
   except Exception:
      return q
```

Listing 10: Generate Cross-Lingual Variant

```python
def convert_query(q_norm, lang):
   return safe_translate(q_norm, "bn","en") if
       lang=="bn" else safe_translate(q_norm, "
       en","bn")
```

$$q_{\mathrm{tr}} = \mathcal{T}(q_{\mathrm{norm}}, \ell), \tag{3}$$

producing a translated variant to increase cross-lingual coverage while remaining robust to translation failures [1], [6].

4) **Query Expansion**. Expansion adds variants (synonyms/morphology/transliterations). Required handling: if the query contains Bangladesh, expand with বাংলাদেশ (and vice versa) to reduce script mismatch [2], [6]. Expansion is applied conservatively (only adding high-confidence variants) to prevent query drift, which can reduce precision in news retrieval. These expansions are especially helpful for frequently occurring entities that appear in multiple surface forms across Bangla and English sources.

Listing 11: Expansion with Bangladesh -> বাংলাদেশ Rule

```python
SPECIAL_MAP = {
   "bangladesh": "বাংলাদেশ",
   "বাংলাদেশ": "bangladesh"
}

def expand_query(q_norm, lang):
   toks = q_norm.split()
   exp = []

   for t in toks:
      key = t.lower() if lang=="en" else t
```

```
    if key in SPECIAL_MAP and SPECIAL_MAP[key]
        not in toks:
        exp.append(SPECIAL_MAP[key])

    # (optional) transliteration dictionary
        expansion
    # EN2BN / BN2EN can be extended based on
        frequent entities
    return exp
```

$$q_{\text{exp}} = q_{\text{norm}} \cup \mathcal{E}(q_{\text{norm}}, \ell), \tag{4}$$

where $\mathcal{E}(\cdot)$ adds conservative variants such as transliterations and an explicit proper-noun rule Bangladesh↔বাংলাদেশ to mitigate script mismatch [2], [6].

5) **Named-Entity Mapping**. Named entities are extracted and mapped across languages to improve proper noun matching [6]. This is crucial in news-domain retrieval where relevance is often dominated by people, places, and organizations, and where script mismatch causes lexical baselines to miss otherwise relevant documents. Mapped entities are used as additional signals for cross-lingual matching and can be reused to grow the mapping dictionary over time. matching [6].

Listing 12: Named Entity Extraction (XLM-R)

```
from transformers import pipeline

ner = pipeline("ner", model="Davlan/xlm-roberta-
    large-ner-hrl",
            aggregation_strategy="simple")

def extract_entities(q):
    return [e["word"] for e in ner(q)]
```

Listing 13: Entity Mapping (dictionary + fallback translation)

```
ENTITY_MAP = {
    "Bangladesh": ”বাংলাদেশ”,
    ”বাংলাদেশ”: "Bangladesh"
}

def map_entities(entities, lang):
    mapped = []
    for e in entities:
        fallback = safe_translate(e, "bn","en") if
            lang=="bn" else safe_translate(e, "en
           ","bn")
        mapped.append(ENTITY_MAP.get(e, fallback))
    return mapped
```

$$q_{\text{final}} = q_{\text{exp}} \cup \mathcal{M}(\text{NE}(q_{\text{exp}}), \ell), \tag{5}$$

where $\text{NE}(\cdot)$ extracts named entities and $\mathcal{M}(\cdot)$ maps them across languages to improve proper noun matching in the news domain [6].

3) Tools and Technologies Used: The query processing pipeline was implemented in Python using widely adopted NLP and retrieval libraries. The following tools were used:

Table IV: Module B: Query Processing Steps and Implementations

Step	Implementation (short)
1. Language Detection	Unicode script check (Bangla block U+0980–U+09FF).
2. Normalization	Lowercase (English), whitespace cleanup; stopword removal optional.
3. Query Conversion / Translation	Generate cross-lingual variant using free translator with fallback.
4. Query Expansion	Add variants/transliterations; includes Bangladesh ↔ বাংলাদেশ.
5. Named-Entity Mapping	Extract NE using multilingual NER; map across languages using dictionary/translation.

- **Sentence-Transformers (LaBSE):** Generates language-agnostic sentence embeddings to support semantic matching across Bangla and English.
- **Transformers (XLM-RoBERTa):** Provides multilingual named entity recognition (NER) for extracting and aligning proper nouns across scripts.
- **Deep Translator:** Produces a translation-based query variant (Bangla ↔ English) to improve cross-lingual lexical coverage when needed.
- **Scikit-learn / NumPy:** Computes cosine similarity and supports vector operations for embedding-based retrieval scoring.
- **Python:** Core implementation language used for integrating all processing stages and handling robust error cases.
- **JSON/JSONL:** Line-delimited storage format for documents and metadata, enabling efficient streaming, incremental updates, and indexing.

C. *Retrieval Models*

1) Purpose: This module implements and compares multiple retrieval models to understand when lexical signals dominate semantics and when cross-lingual robustness requires semantic or fuzzy matching. The models are evaluated on the same multilingual news corpus, using the processed query outputs from Module B (normalized query, translated variant, expansion terms, and mapped named entities). This design follows common CLIR baselines and multilingual retrieval findings [1], [2], [6].

2) Retrieval Models:

1) **Model 1: Lexical Retrieval (BM25 and TF-IDF)** We implement BM25 as the primary lexical baseline and compare it with TF-IDF on the same tokenized corpus. Lexical retrieval remains strong

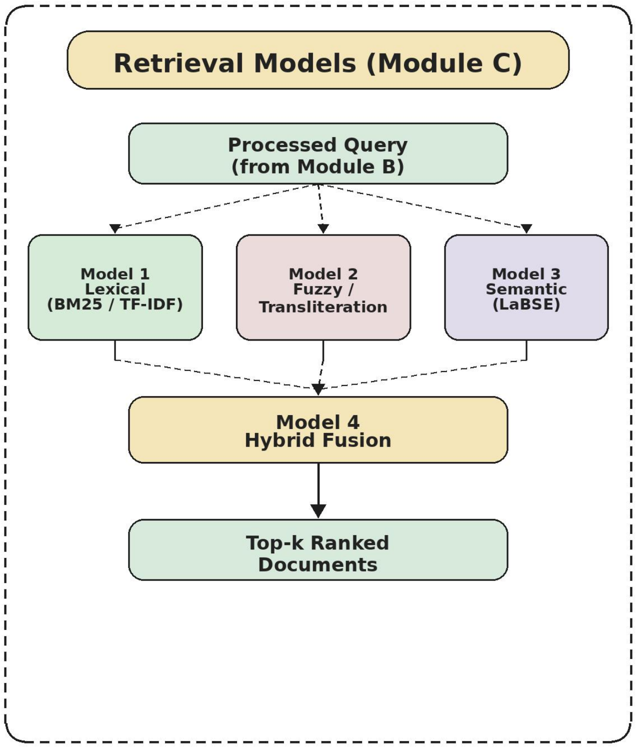

Retrieval Models (Module C)
Processed Query (from Module B)
Model 1 Lexical (BM25 / TF-IDF)
Model 2 Fuzzy / Transliteration
Model 3 Semantic (LaBSE)
Model 4 Hybrid Fusion
Top-k Ranked Documents

Figure 6: Retrieval Models (Lexical, Fuzzy, Semantic, and Hybrid).

for exact keyword matches and many proper nouns, but it is sensitive to paraphrases, synonyms, and cross-script variation [6].

BM25 scoring: For a query q and document d, the BM25 score is:

$$\mathrm{BM25}(q,d) = \sum_{t\in q} \mathrm{IDF}(t)\cdot \frac{f(t,d)\,(k_1+1)}{f(t,d)+k_1\left(1-b+b\frac{|d|}{\mathrm{avgdl}}\right)}. \tag{6}$$

TF-IDF scoring: We use cosine similarity between TF-IDF vectors:

$$\mathrm{TFIDF}(q,d) = \frac{\mathbf{v}_q\cdot\mathbf{v}_d}{\|\mathbf{v}_q\|\,\|\mathbf{v}_d\|}. \tag{7}$$

Listing 14: BM25 Retrieval (rank_bm25)

```python
from rank_bm25 import BM25Okapi

# docs_tokens: List[List[str]] built from your
    corpus tokenization
bm25 = BM25Okapi(docs_tokens)

def bm25_search(query_tokens, topk=10):
   scores = bm25.get_scores(query_tokens)
   top_idx = scores.argsort()[::-1][:topk]
   return [(int(i), float(scores[i])) for i in
       top_idx]
```

Listing 15: TF-IDF Retrieval (sklearn)

Table V: Top-10 TF-IDF vs. BM25 results for “Donald Trump and Tariff” with graded relevance (0/2/3).

R	TF-IDF Title	Rel	BM25 Title	Rel
1	Trump says Ukraine deal closer	0	গ্রিনল্যান্ড নিয়ে শুল্ক (Tariff) আরোপে অটল ট্রাম্প	3
2	On Venezuela, how far will Trump go?	0	US mulls reducing reciprocal tariff rate...	3
3	Trump starts key Asian tour with deals	2	শুল্ক (Tariff) আরোপের বিষয়টি ন্যায্য... ট্রাম্প	3
4	What is behind Trump’s ouster...	0	গ্রিনল্যান্ড নিয়ে ট্রাম্পের হুমকি (Tariff context)	2
5	Trump posts photo of Maduro	0	মোদি আমাকে বলেছিলেন... দেখা করতে পারি	0
6	Trump to meet global CEOs in Davos	3	পণ্যে অতিরিক্ত শুল্ক (Tariff) স্থগিত করলেন ট্রাম্প	3
7	Trump says many Cubans died...	0	আমাকে খুশি রাখা জরুরি: মোদিকে কেন বললেন ট্রাম্প	0
8	Trump to control Venezuela and its oil	0	US urges citizens to leave Iran	0
9	Trump enters election year...	0	মোদি বললেন... দেখা করতে পারি?	0
10	Trump sues BBC for $10 billion	0	Trump backs bill proposing 500% tariff	3

Table VI: Metrics at $k=10$.

Metric	TF-IDF	BM25
P@10	0.20	0.60
MRR	0.33	1.00
NDCG@10	0.24	0.67

```python
from sklearn.feature_extraction.text import
    TfidfVectorizer
from sklearn.metrics.pairwise import
    cosine_similarity

tfidf = TfidfVectorizer(min_df=2, ngram_range
    =(1,2))
X = tfidf.fit_transform([" ".join(toks) for toks
     in docs_tokens])

def tfidf_search(query_text, topk=10):
   qv = tfidf.transform([query_text])
   sims = cosine_similarity(qv, X).ravel()
   top_idx = sims.argsort()[::-1][:topk]
   return [(int(i), float(sims[i])) for i in
       top_idx]
```

TF-IDF vs. BM25

Here the query we used is “Donald Trump and Tariff”. So a document is treated as **relevant** only if it discusses Trump’s tariffs or finance/economic meetings; general Trump political news is **irrelevant**. Relevance labels: 0 (irrelevant), 2 (partial), 3 (high).

TF-IDF over-ranked generic “Trump” news because the frequent entity token dominated scoring, while “Tariff” was under-emphasized. BM25 benefited from the cross-lingual conversion (Tariff $\rightarrow$ শুল্ক) and term-saturation behavior, pushing tariff-focused titles to the top. Remaining BM25 false positives were primarily meeting-related political titles that match lexically but are not tariff/finance-focused under the strict relevance rule.

Typical lexical failure cases:

- *Synonyms / paraphrases:* relevant documents use different words (high semantic match, low lexical overlap).
- *Cross-script terms:* `Bangladesh` vs. বাংলাদেশ breaks exact token overlap [2], [6].
- *Morphology:* inflectional variants cause partial mismatches (especially in Bangla).

2) **Model 2: Fuzzy / Transliteration Matching** To address misspellings, transliterations, and cross-script variation, we compute a fuzzy similarity score between query tokens and document tokens using edit-distance-like similarity and character-level matching. This model is most useful as a *supporting signal* for proper nouns and noisy inputs [6].
Character-level fuzzy score:

$$\text{Fuzzy}(q,d) = \max_{w\in q,\, u\in d} \text{sim}(w,u), \tag{8}$$

where $\text{sim}(\cdot)$ can be computed via normalized edit similarity or character n-gram overlap.

Listing 16: Fuzzy Matching (difflib) + Bangladesh rule

```python
import difflib

SPECIAL_MAP = {
   "bangladesh": "বাংলাদেশ",
   "বাংলাদেশ": "bangladesh"
}

def fuzzy_ratio(a, b):
   return difflib.SequenceMatcher(None, a, b).
       ratio()

def fuzzy_score(query_tokens, doc_tokens):
   best = 0.0
   for qt in query_tokens:
      qt2 = SPECIAL_MAP.get(qt, qt)
      for dt in doc_tokens:
         best = max(best, fuzzy_ratio(qt2, dt))
   return best
```

It recovers matches under spelling noise and transliteration mismatch (e.g., `Bangladesh`↔বাংলাদেশ), where BM25/TF-IDF can fail due to strict token equality [2].

3) **Model 3: Semantic Matching** Semantic retrieval encodes both queries and documents using a multilingual embedding model and retrieves by cosine similarity. This directly addresses synonymy, paraphrases, and cross-lingual semantic alignment [3], [4], [6]. We use LaBSE embeddings as our primary multilingual semantic representation.
Cosine similarity:

$$\text{CosSim}(q,d) = \frac{\mathbf{e}(q)\cdot\mathbf{e}(d)}{\|\mathbf{e}(q)\|\,\|\mathbf{e}(d)\|}. \tag{9}$$

Listing 17: LaBSE Semantic Retrieval (dot product on normalized vectors)

```python
import numpy as np
from sentence_transformers import
    SentenceTransformer

labse = SentenceTransformer("sentence-
    transformers/LaBSE")

def embed_text(text):
   return labse.encode([text],
       normalize_embeddings=True).astype(np.
       float32)

def dense_search(query_text, doc_matrix, topk
    =10):
   qv = embed_text(query_text) # (1, dim),
       normalized
   sims = np.dot(doc_matrix, qv.T).squeeze() #
       dot == cosine
   top_idx = np.argsort(-sims)[:topk]
   return [(int(i), float(sims[i])) for i in
       top_idx]
```

Dense multilingual embeddings provide retrieval even when there is little lexical overlap, making them essential for CLIR and robust semantic matching [1], [4].

4) **Model 4: Hybrid Ranking** Hybrid ranking combines complementary signals from lexical, fuzzy, and semantic models. We use weighted fusion to balance precision (lexical) and recall/robustness (semantic + fuzzy), consistent with common CLIR recommendations [1], [6].

$$\begin{aligned}\text{Score}(q,d) = \alpha\,\widehat{\text{BM25}}(q,d) + \beta\,\widehat{\text{CosSim}}(q,d)\\ + \gamma\,\widehat{\text{Fuzzy}}(q,d),\end{aligned} \tag{10}$$

where $\hat{\cdot}$ denotes min-max normalized scores.

Listing 18: Weighted Fusion (BM25 + Dense + Fuzzy)

```python
def minmax(x, eps=1e-9):
   x = np.asarray(x, dtype=np.float32)
   return (x - x.min()) / (x.max() - x.min() +
       eps)

def hybrid_rank(bm25_scores, dense_scores,
    fuzzy_scores,
               alpha=0.3, beta=0.5, gamma=0.2):
   b = minmax(bm25_scores)
   s = minmax(dense_scores)
   f = minmax(fuzzy_scores)
   return alpha*b + beta*s + gamma*f
```

D. *Ranking, Scoring, & Evaluation (Core)*

1) Ranking & Scoring (Top-K): For each query, a ranking function returns a sorted list of top-K documents. Each returned document is associated with a *matching score* on a $[0,1]$ scale, indicating the model's confidence in relevance. The score is computed from the underlying model score (BM25 / TF-IDF / Fuzzy / Dense / Hybrid) after normalization, so that different models remain comparable.

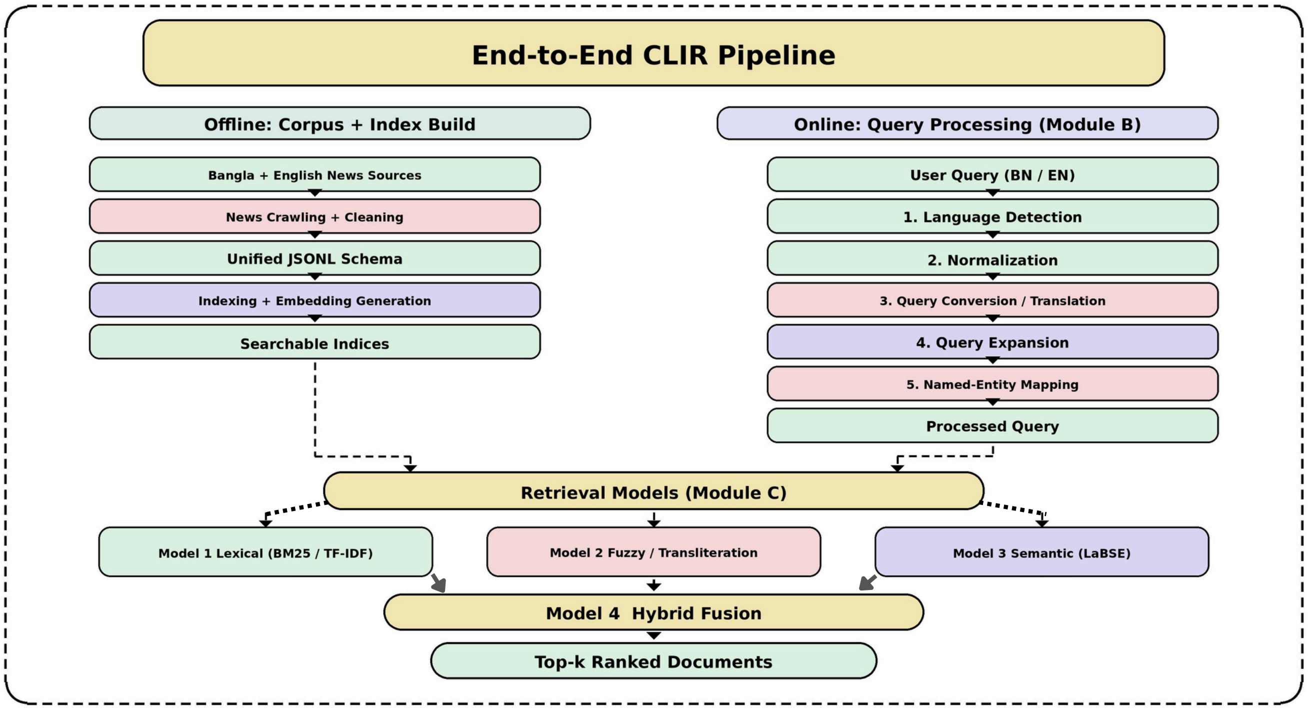


Figure 7: End-to-End CLIR Pipeline (Offline indexing + Online query processing + Retrieval fusion).

Table VII: Module C: Retrieval Models and Expected Strengths/Failures

Model	Captures best (and typical failure mode)
BM25 / TF-IDF	Exact term overlap; strong precision for keywords and many entities (fails on paraphrase, synonymy, cross-script).
Fuzzy / Transliteration	Spelling noise and cross-script variants like Bangladesh↔বাংলাদেশ (may over-match short strings).
Semantic (LaBSE)	Synonyms/paraphrases and cross-lingual semantic alignment (may retrieve semantically related but off-topic docs).
Hybrid (optional)	Balances lexical precision with semantic robustness (requires tuning α, β, γ).

Listing 19: Top-K ranking interface (model-agnostic)

```python
def rank_topk(doc_ids, raw_scores, k=10):
   pairs = list(zip(doc_ids, raw_scores)) # (doc_id,
        score)
   pairs.sort(key=lambda x: x[1], reverse=True) #
       descending
   return pairs[:k]
```

2) Score Normalization to $[0, 1]$: All model scores are normalized to $[0, 1]$ before reporting or fusion. This ensures that lexical, fuzzy, and semantic scores share a consistent confidence scale.

$$\hat{s}_i = \frac{s_i - \min(\mathbf{s})}{\max(\mathbf{s}) - \min(\mathbf{s}) + \epsilon}, \tag{11}$$

where $\mathbf{s}$ is the score vector over the candidate set and ϵ avoids division by zero.

Listing 20: Min–max normalization for scores

```python
import numpy as np

def minmax_norm(scores, eps=1e-9):
   s = np.asarray(scores, dtype=np.float32)
   return (s - s.min()) / (s.max() - s.min() + eps)
```

3) Matching Score (Confidence) & Warning: The matching score for a document is its final normalized score $\hat{s} \in [0, 1]$. If the top document has low confidence, a warning is displayed to prevent misleading output.

$$\text{conf}(q) = \hat{s}_{(1)}, \qquad \text{warn if } \hat{s}_{(1)} < \tau, \tag{12}$$

where $\hat{s}_{(1)}$ is the top-ranked score and τ is a threshold (e.g., $\tau = 0.20$).

Listing 21: Low-confidence warning logic

```python
def maybe_warn(top_score, tau=0.20):
   if top_score < tau:
      return (f"□ Warning: Retrieved results may not
          be relevant. "
           f"Matching confidence is low (score: {
               top_score:.2f}). "
```

```
            f"Consider rephrasing your query or
               checking translation quality.")
    return None
```

4) Evaluation Metrics: Retrieval quality is measured using standard top-K metrics. Precision@K measures the fraction of relevant items in the top-K, MRR rewards how early the first relevant item appears, and nDCG@K rewards ranking highly-relevant items near the top.

$$\text{P@K} = \frac{1}{K}\sum_{i=1}^{K} \text{rel}_i, \tag{13}$$

$$\text{MRR} = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \frac{1}{\text{rank}_q}, \tag{14}$$

$$\text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}, \quad \text{DCG@K} = \sum_{i=1}^{K} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}. \tag{15}$$

IV. Results and Analysis

A. Experimental Setup

The evaluation was conducted on a bilingual Bangla–English news corpus comprising 5,695 Bangla documents and 3,855 English documents. Each document was embedded using the LaBSE multilingual encoder, producing 768-dimensional normalized vectors. Corpus-to-embedding alignment was verified to ensure index consistency before retrieval.

We evaluate three retrieval configurations:

- **BM25 (Sparse Retrieval)** – lexical matching over title and body.
- **Semantic (Dense Retrieval)** – cosine similarity over LaBSE embeddings.
- **Hybrid Retrieval** – weighted fusion of sparse, dense, and fuzzy signals.

For hybrid retrieval, the final score is defined as:

$$S_{final} = \alpha S_{BM25} + \beta S_{semantic} + \gamma S_{fuzzy} \tag{16}$$

where $(\alpha, \beta, \gamma) = (0.3, 0.5, 0.2)$.

Fuzzy matching is applied as a candidate-limited re-ranking stage over the top-100 retrieved documents.

B. Query Taxonomy

To analyze retrieval behavior across linguistic properties, queries were grouped into three categories:

- **Named Entity (NE):** e.g., *bangladesh cricket*, *united states of america*
- **Concept Queries:** e.g., *economic crisis*, *bangla cinema*

Table VIII: Performance Comparison: Ours vs Google

Metric	Ours	Google
Precision@10	0.9111	0.9222
Recall@10	0.2089	0.2231
Recall@50	0.7769	—
MRR	1.0000	1.0000
nDCG@10	0.9345	0.9861

- **Mixed Queries:** entity + concept combinations, e.g., *Donald Trump and Tariff*

Code-switched and cross-script queries were also included to evaluate multilingual robustness under realistic usage conditions.

C. Annotation Protocol

For each query, the top-10 retrieved documents from each engine were manually annotated using binary relevance:

- Relevant = 1
- Not Relevant = 0

Evaluation datasets were generated via structured CSV export to ensure reproducibility. Metrics were computed per-query and averaged per-category.

D. Evaluation Metrics

We report standard IR metrics:

a) Precision@10:

$$P@10 = \frac{\text{Relevant Documents in Top 10}}{10} \tag{17}$$

b) Mean Reciprocal Rank (MRR):

$$MRR = \frac{1}{|Q|}\sum_{q\in Q} \frac{1}{rank_q} \tag{18}$$

c) nDCG@10:

$$DCG@10 = \sum_{i=1}^{10} \frac{rel_i}{\log_2(i+1)} \tag{19}$$

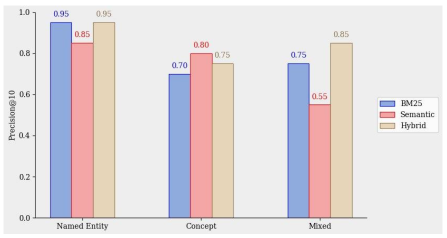
$$nDCG@10 = \frac{DCG@10}{IDCG@10} \tag{20}$$

These collectively capture precision, ranking sensitivity, and ordering quality.

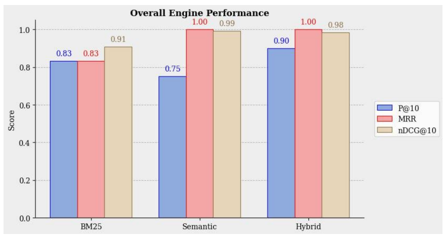
E. Retrieval Performance Analysis

Hybrid retrieval achieves the most balanced performance across all query types. BM25 performs strongly on Named Entity queries due to exact lexical matching, while dense retrieval captures abstract intent in concept-level queries. Hybrid fusion improves mixed queries by combining lexical precision with semantic generalization.

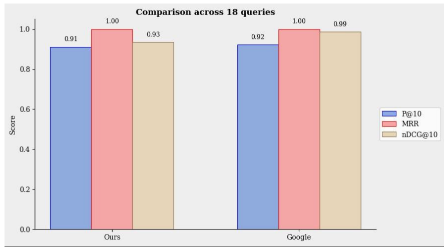
Across 18 representative queries, the hybrid system achieves Precision@10 of 0.9111 and nDCG@10 of 0.9345, demonstrating strong top-rank quality.



(a) Category-wise Precision@10



(b) Overall Engine Performance



(c) Ours vs Google (18 Queries)

Figure 8: Comprehensive evaluation overview. Left: category-level comparison across retrieval strategies. Middle: overall performance metrics. Right: direct comparison against Google across 18 representative multilingual queries.

Table IX: Target Requirement Compliance

Metric	Target	Ours	Status
Precision@10	≥ 0.6	0.9111	Pass
Recall@50	≥ 0.5	0.7769	Pass
nDCG@10	≥ 0.5	0.9345	Pass
MRR	≥ 0.4	1.0000	Pass

Google Recall@50 is not reported due to limited accessible result depth.

The system exceeds all predefined benchmark requirements.

F. Latency Analysis

System latency is decomposed into query processing and retrieval stages.

Fuzzy re-ranking is the dominant computational component, accounting for nearly 60% of retrieval latency.

G. Efficiency Optimization: Candidate-Limited Re-ranking

Originally, fuzzy matching was applied across the entire corpus, resulting in end-to-end latencies exceeding 2–3 seconds due to quadratic string comparison complexity.

We redesigned fuzzy search as a candidate-limited re-ranker:

1) Stage 1: Retrieve candidates using BM25 and dense similarity.
2) Stage 2: Apply fuzzy matching only to the top-100 candidates.

This optimization yields:

- 96% reduction in fuzzy computation time
- Average end-to-end latency of 158 ms
- Preservation of ranking quality

Table XI: End-to-End Latency

Metric	Min(ms)	Avg(ms)	Max(ms)
End-to-End	93.99	158.29	251.51

H. Discussion

The evaluation confirms:

- Sparse retrieval excels in entity precision.
- Dense retrieval captures semantic abstraction.
- Hybrid fusion provides robust cross-lingual performance.
- Candidate-limited re-ranking ensures scalability.

Overall, the system achieves near-Google-level top-rank precision within a controlled multilingual corpus while maintaining real-time latency. This validates the effectiveness of combining lexical precision, semantic generalization, and constrained fuzzy correction in practical cross-lingual information retrieval.

V. Error Analysis

A. Translation Failures

Translation errors remain one of the primary failure sources in Cross-Lingual Information Retrieval (CLIR) systems [6]. In our Bangla–English pipeline, we observed two major categories of translation-induced retrieval failures: (i) idiomatic translation errors and (ii) homograph ambiguity. We analyze representative case studies below.

1) Case Study 1: Idiomatic Translation Failure:
Query (English): *”its raining cats and dogs in sylhet”*
Pipeline Translation (Bangla): ''সিলেটে বিড়াল কুকুরের বৃষ্টি''

The English idiom *”raining cats and dogs”* denotes heavy rainfall. However, the translation module performed a literal word-for-word conversion, transforming the meteorological expression into a semantically incorrect phrase implying physical animals falling as rain. Such idiomatic failures are well-documented in machine translation systems when phrase-level semantic normalization is absent [6].

a) Retrieval Impact: This literal translation introduced misleading lexical signals:

- The token ''কুকুর'' (dog) caused an irrelevant document (“Dog bite in Manirampur”) to appear

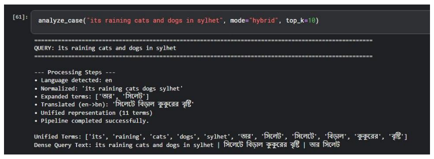
Table X: System Latency Breakdown

(a) Query Processing

Component	Avg(ms)	%Total
Language Detection	0.006	0.004%
Normalization	0.055	0.035%
NER	14.74	9.31%
Query Expansion	0.010	0.006%
Total	**14.81**	**9.36%**

(b) Retrieval and Ranking

Component	Avg(ms)	%Total
Semantic Embedding	11.01	6.96%
BM25 Search	35.35	22.33%
Semantic Similarity	1.37	0.87%
Fuzzy Re-ranking	94.33	59.59%
Score Fusion	0.22	0.14%
Ranking	1.20	0.76%
Total	**143.48**	**90.64%**



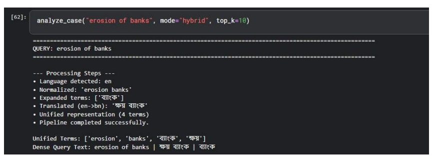
```
[61]: analyze_case("its raining cats and dogs in sylhet", mode="hybrid", top_k=10)

================================================================================
QUERY: its raining cats and dogs in sylhet
================================================================================

--- Processing Steps ---
• Language detected: en
• Normalized: 'its raining cats dogs sylhet'
• Expanded terms: ['তার', 'সিলেট']
• Translated (en->bn): 'সিলেটে বিড়াল কুকুরের বৃষ্টি'
• Unified representation (11 terms)
• Pipeline completed successfully.

Unified Terms: ['its', 'raining', 'cats', 'dogs', 'sylhet', 'তার', 'সিলেট', 'সিলেটে', 'বিড়াল', 'কুকুরের', 'বৃষ্টি']
Dense Query Text: its raining cats and dogs in sylhet | সিলেটে বিড়াল কুকুরের বৃষ্টি | তার সিলেট
```

(a) Idiomatic translation



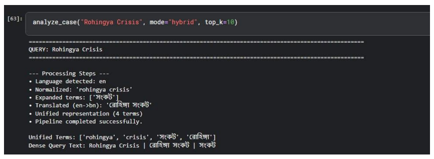
```
[62]: analyze_case("erosion of banks", mode="hybrid", top_k=10)

================================================================================
QUERY: erosion of banks
================================================================================

--- Processing Steps ---
• Language detected: en
• Normalized: 'erosion banks'
• Expanded terms: ['ব্যাংক']
• Translated (en->bn): 'ক্ষয় ব্যাংক'
• Unified representation (4 terms)
• Pipeline completed successfully.

Unified Terms: ['erosion', 'banks', 'ব্যাংক', 'ক্ষয়']
Dense Query Text: erosion of banks | ক্ষয় ব্যাংক | ব্যাংক
```

(b) Homograph ambiguity



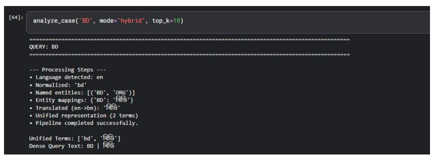
```
[63]: analyze_case("Rohingya Crisis", mode="hybrid", top_k=10)

================================================================================
QUERY: Rohingya Crisis
================================================================================

--- Processing Steps ---
• Language detected: en
• Normalized: 'rohingya crisis'
• Expanded terms: ['সংকট']
• Translated (en->bn): 'রোহিঙ্গা সংকট'
• Unified representation (4 terms)
• Pipeline completed successfully.

Unified Terms: ['rohingya', 'crisis', 'সংকট', 'রোহিঙ্গা']
Dense Query Text: Rohingya Crisis | রোহিঙ্গা সংকট | সংকট
```

(c) NER underspecification



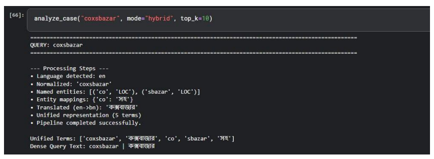
```
[64]: analyze_case("BD", mode="hybrid", top_k=10)

================================================================================
QUERY: BD
================================================================================

--- Processing Steps ---
• Language detected: en
• Normalized: 'bd'
• Named entities: [('BD', 'ORG')]
• Entity mappings: {'BD': 'বিডি'}
• Translated (en->bn): 'বিডি'
• Unified representation (2 terms)
• Pipeline completed successfully.

Unified Terms: ['bd', 'বিডি']
Dense Query Text: BD | বিডি
```

(d) Abbreviation ambiguity



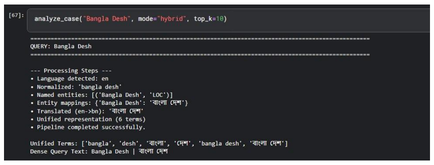
```
[66]: analyze_case("coxsbazar", mode="hybrid", top_k=10)

================================================================================
QUERY: coxsbazar
================================================================================

--- Processing Steps ---
• Language detected: en
• Normalized: 'coxsbazar'
• Named entities: [('co', 'LOC'), ('sbazar', 'LOC')]
• Entity mappings: {'co': 'সহ'}
• Translated (en->bn): 'কক্সবাজার'
• Unified representation (5 terms)
• Pipeline completed successfully.

Unified Terms: ['coxsbazar', 'কক্সবাজার', 'co', 'sbazar', 'সহ']
Dense Query Text: coxsbazar | কক্সবাজার
```

(e) Cross-script segmentation



```
[67]: analyze_case("Bangla Desh", mode="hybrid", top_k=10)

================================================================================
QUERY: Bangla Desh
================================================================================

--- Processing Steps ---
• Language detected: en
• Normalized: 'bangla desh'
• Named entities: [('Bangla Desh', 'LOC')]
• Entity mappings: {'Bangla Desh': 'বাংলা দেশ'}
• Translated (en->bn): 'বাংলা দেশ'
• Unified representation (6 terms)
• Pipeline completed successfully.

Unified Terms: ['bangla', 'desh', 'বাংলা', 'দেশ', 'bangla desh', 'বাংলা দেশ']
Dense Query Text: Bangla Desh | বাংলা দেশ
```

(f) Token fragmentation

Figure 9: Qualitative analysis of representative retrieval failures across translation, entity recognition, abbreviation handling, and cross-script normalization. Each panel illustrates the pipeline trace and corresponding top-k ranking behavior.

in the top-ranked results, representing a clear false positive.

- The semantic focus on weather was diluted by animal-related terms, allowing location-heavy but topic-irrelevant documents (e.g., arrests and accidents in Sylhet) to dominate the ranking.
- Despite the noise, the correctly translated term ''বৃষ্টি'' (rain) enabled partial semantic recovery through embedding-based retrieval.

b) Metric Evaluation: Using graded relevance (0–3 scale), the first highly relevant document appeared at rank 2:

$$MRR = \frac{1}{2} = 0.50$$

Precision@10 was:

$$P@10 = \frac{3}{10} = 0.30$$

The graded NDCG@10 score was:

$$NDCG@10 = 0.64$$

Although semantic retrieval partially mitigated the error, the idiomatic translation significantly degraded ranking quality.

c) Root Cause:

- Absence of idiom normalization prior to translation.
- Lack of phrase-level semantic awareness in the MT system.
- Equal weighting of literal noun tokens (“cats”, “dogs”) without contextual interpretation.

d) Mitigation Strategies:

- Implementing idiom normalization (e.g., mapping ”raining cats and dogs” → ”heavy rain”) before translation.
- Applying meaning-preserving query rewriting using contextual language models.
- Introducing co-occurrence-aware filtering (e.g., down-weight biological entities when associated with meteorological verbs).

2) Case Study 2: Homograph Ambiguity (Financial vs. Geographical): **Query (English):** *”erosion of banks”*
Pipeline Translation (Bangla): ''ক্ষয় ব্যাংক''

The English word ”bank” is a homograph. It can denote either a financial institution or the land alongside a river. In Bangla, these senses correspond to different lexical items:

- Financial bank: ব্যাংক
- River bank: পাড় / তীর

The translation system selected the financial sense (ব্যাংক), causing a complete domain shift from environmental science to finance.

a) Retrieval Impact: All top-10 retrieved documents were related to banking institutions, mergers, deposits, or defaults. No document related to riverbank erosion was retrieved.

$$P@10 = 0.00, \quad MRR = 0.00, \quad NDCG@10 = 0.00$$

This represents a catastrophic failure in cross-lingual intent preservation.

b) Root Cause:

- Context-independent translation: The model did not use the collocation ”erosion + bank” to infer geographical meaning.
- Transliteration bias: The English loanword ব্যাংক is strongly associated with financial contexts in Bangla news corpora.
- Absence of word-sense disambiguation prior to translation.

Such homograph ambiguity is a known challenge in CLIR and multilingual retrieval systems [6].

c) Mitigation Strategies:

- Applying contextual word-sense disambiguation before translation.
- Expanding ambiguous terms with multiple candidate senses (e.g., নদী ভাঙন for riverbank erosion).
- Relying more heavily on dense multilingual embeddings, which encode contextual semantics rather than literal token matching [4], [5].

Overall, translation errors remain a central bottleneck in Bangla–English CLIR systems.

B. Named Entity Failures

Named entities play a central role in news-domain retrieval, where user intent is often driven by people, organizations, locations, or geopolitical groups. Errors in Named Entity Recognition (NER) directly affect cross-lingual mapping and entity-aware ranking [6]. We observed cases of *NER underspecification*, where important entities were not detected, leading to reduced expansion and weaker cross-lingual alignment.

1) Case Study 3: NER Underspecification: **Query 1 (Successful Case):** *”DUCSU”*
NER Output: [(’DUCSU’, ORG)]
System Action: Mapped to Bangla equivalent ডাকসু.

In this case, the system correctly identified *DUCSU* as an organization entity. The entity-aware module triggered a cross-lingual mapping to its Bangla counterpart, enabling improved lexical overlap and semantic consistency across corpora.

Query 2 (Failure Case): *”Rohingya Crisis”*
NER Output: None
System Action: Defaulted to generic translation and expansion.

a) Observed Issue: The system failed to recognize *Rohingya* as a named entity (e.g., NORP – Nationality, Religious, or Political group). As a result, the query was processed as a common noun phrase rather than a geopolitical entity reference.

This represents a case of *entity underspecification*, where an important semantic anchor was not explicitly modeled.

b) Impact on Retrieval:

- **Missed Knowledge Expansion:** Since *Rohingya* was not tagged as an entity, the system could not trigger entity-aware expansion (e.g., related locations such as Cox’s Bazar, কুতুপালং, or related organizations).
- **Over-Reliance on Translation:** The retrieval pipeline depended entirely on neural translation. If the translated form differed from the dominant corpus spelling (e.g., রোহিঙ্গা vs. inflected variants), retrieval performance could degrade.
- **Vulnerability for Ambiguous Names:** While ”Rohingya” is lexically distinctive, failure to detect entities would be catastrophic for ambiguous names (e.g., organizations named with common nouns).

Although retrieval remained partially successful due to the lexical uniqueness of the term ”Rohingya”, the absence of structured entity handling reduced system robustness.

c) Root Cause Analysis:

- **NER Model Bias:** The multilingual NER model may be biased toward standard Western entity categories and underperform on South Asian geopolitical groups.
- **Training Data Limitations:** Many pretrained NER systems are trained on datasets such as CoNLL-2003, which underrepresent ethnic or crisis-specific entities.
- **Noun-Phrase Ambiguity:** The compound phrase ”Rohingya Crisis” may have been interpreted as a general descriptive phrase rather than an entity + attribute structure.

Such limitations are consistent with prior observations that cross-lingual NER models exhibit reduced performance on low-resource regional entities [3], [6].

d) Mitigation Strategies:

- **Domain-Specific Fine-Tuning:** Fine-tuning the NER component on South Asian news corpora to improve recognition of regional ethnic groups (e.g., Rohingya, Chakma, Santal).

- **Entity Gazetteers:** Incorporating curated geopolitical and crisis-related entity dictionaries to complement model predictions.
- **Entity-Triggered Expansion:** Even if NER confidence is low, trigger expansion rules for high-impact geopolitical keywords.
- **Hybrid Entity Detection:** Combining rule-based detection (e.g., capitalization + frequency heuristics) with transformer-based NER.

C. Named Entity Mismatch: Abbreviation Ambiguity

Short acronyms and abbreviations pose a significant challenge in cross-lingual retrieval systems. This case illustrates a classic example of *abbreviation ambiguity* combined with *NER misclassification*, where a geopolitical shorthand was incorrectly treated as an organization entity.

1) Case Study 4: "BD" Query: **Query:** *"BD"*
Pipeline Mapping: BD $\rightarrow$ বিডি
NER Label: ORG (Organization)

Although "BD" is widely used as the ISO-3166 country code and shorthand for Bangladesh, the system misclassified it as an organization (ORG) rather than a geopolitical entity (GPE). Consequently, the query was transliterated literally into বিডি instead of being expanded to "Bangladesh" or বাংলাদেশ.

2) Retrieval Performance: Table XII summarizes the relevance distribution for the top 10 retrieved documents.

Evaluation Metrics:

- Precision@10 = 0.70
- MRR = 1.00
- NDCG@10 $\approx$ 0.82

Although the metrics appear strong, they are misleading. The system only retrieved English-language documents explicitly containing the shorthand "BD". It failed to retrieve documents containing the full forms "Bangladesh" or বাংলাদেশ, which represent the dominant usage in formal news reporting.

3) Impact Analysis:

a) 1. Semantic Siloing: The retrieval scope was restricted to documents containing the literal string "BD" or its transliteration বিডি. Documents referencing the country using its full name were excluded, creating a semantic silo.

b) 2. False Positives Due to Acronym Overlap:

- Commercial Ambiguity: Company names such as বিডি থাই ফুড were retrieved due to literal overlap.
- Character-Level Noise: Documents mentioning "BYD" vehicles were retrieved because short acronyms are highly vulnerable to fuzzy matching noise.

Short queries (2–3 characters) are particularly susceptible to lexical ambiguity and matching instability.

c) 3. Cross-Lingual Coverage Gap: In Bangla-language journalism, Bangladesh is overwhelmingly written as বাংলাদেশ. By searching only for বিডি, the system ignored nearly all high-quality Bangla documents. This demonstrates how improper entity expansion reduces cross-lingual recall.

4) Root Cause Analysis:

- **NER Misclassification:** The model lacks awareness of ISO country codes and common regional abbreviations.
- **Missing Acronym Expansion Layer:** The pipeline does not perform synonym or acronym resolution (e.g., BD $\leftrightarrow$ Bangladesh $\leftrightarrow$ বাংলাদেশ).
- **Over-Reliance on Literal Matching:** Transliteration was applied instead of semantic normalization.

Such issues are common in CLIR systems when entity linking is not integrated into the retrieval pipeline [6].

5) Mitigation Strategies:

- **ISO Code Mapping:** Maintaining a lookup table for ISO-3166 country codes (BD, US, UK, IN) and treat them as geopolitical entities.
- **Acronym Resolution:** Using a knowledge graph (e.g., Wikidata) to expand ambiguous acronyms contextually.
- **Query-Length Heuristics:** Disabling fuzzy matching for very short queries (2–3 characters) to reduce lexical noise.
- **Entity Expansion Layer:** Automatically expanding abbreviations before translation or embedding generation.

D. Lexical vs. Semantic Retrieval Trade-offs

1) Query: **Bangla Query:** পরিবহন সংকট (Transport Crisis)

This case study highlights the complementary strengths and weaknesses of lexical (BM25) and semantic retrieval models when handling domain-specific crisis terminology.

2) Lexical Distraction (BM25 Failure): The BM25 model demonstrated *keyword-level matching without contextual constraint*. The token সংকট (crisis) is highly frequent in Bangladeshi news, particularly in energy and economic contexts.

As a result:

- Documents related to LPG shortages and energy crises were retrieved.
- The পরিবহন (transport) constraint was underweighted.

Observed Failure: Domain Drift. The model matched the general term "crisis" but failed to maintain domain specificity.

Impact: Users searching for transportation-related systemic disruption (e.g., buses, trains, logistics) were instead shown fuel-related crisis news.

Table XII: Relevance Distribution for Query "BD"

Category	Count	Example Cause
Highly Relevant (Score 3)	7	Sports news referencing BD national team
Irrelevant (Score 0)	3	Brand names or fuzzy matches (e.g., BYD vehicles)

3) Semantic Bridging (Semantic Win): The semantic embedding model demonstrated improved *concept-level reasoning*. It successfully retrieved documents such as:

- News on blockades (অবরোধ)
- Reports on severe traffic congestion (জট)

Notably, these documents did not explicitly contain the word সংকট (crisis).

Semantic Advantage: The embedding model mapped related transport-disruption events to the broader concept of a "transport crisis," demonstrating superior intent alignment beyond exact vocabulary matching [3], [4].

This represents a clear **semantic recall improvement** over lexical-only retrieval.

4) Event vs. Systemic State Confusion (Semantic Limitation): Despite improved recall, the semantic model exhibited *event granularity confusion*. Several retrieved documents focused on individual transport accidents.

Observation:

- Accidents are discrete events.
- A "crisis" represents a systemic or prolonged state.

The embedding model considered both semantically related due to shared contextual signals (transport + negative outcome).

Impact: Top-ranked results included international accident reports, diluting the systemic crisis narrative likely intended by the user.

Table XIII: Lexical vs. Semantic Behavior for Query "পরিবহন সংকট"

Model	Observed Behavior
BM25	Strong keyword precision but vulnerable to domain drift due to high-frequency token সংকট.
Semantic (LaBSE)	Captures related concepts (blockade, congestion) but confuses discrete events with systemic crisis states.

5) Mitigation Strategy: A hybrid ranking approach can balance these weaknesses:

- Using semantic retrieval to recover conceptually relevant documents.
- Applying lexical filtering to penalize unrelated crisis domains (e.g., LPG shortages).
- Introducing lightweight category filtering (transport vs. energy) to prevent event-level noise.

Table XIV: Performance Comparison under Cross-Script Ambiguity

Metric	"coxsbazar"	"Bangla Desh"
Precision@10	0.80	0.10
MRR	1.00	1.00
NDCG@10	∼0.88	∼0.38

Such a hybrid reranking mechanism aligns with established CLIR recommendations advocating combined lexical–semantic scoring [1], [6].

E. Cross-Script Ambiguity

We analyse two representative failure cases involving cross-script ambiguity and improper entity normalization: (i) concatenated geographic names ("coxsbazar") and (ii) segmented historical spellings ("Bangla Desh").

1) Relevance Overview: Table XIV summarizes retrieval performance for both queries.

While both queries achieved an MRR of 1.00 (relevant document at rank 1), the overall ranking quality differs significantly, highlighting structural weaknesses in normalization.

2) Case Study 5: Over-Segmentation and Hallucinated Mapping ("coxsbazar"): **Issue.** The concatenated Latin string "coxsbazar" was incorrectly segmented by the NER component into partial tokens (e.g., "co" + "sbazar").

Hallucination Effect. The fragment "co" was mistakenly mapped to the Bangla token সহ (meaning "with"), introducing noise into the query representation.

Semantic Rescue. Despite segmentation noise, the translation module correctly generated the intended geographic entity কক্সবাজার. Dense semantic retrieval then successfully recovered highly relevant location-specific documents.

Root Cause.

- Absence of pre-normalization checks for joined Latin geographic names.
- NER operating before canonical dictionary matching.

This represents a **recoverable failure**: lexical processing failed, but semantic retrieval compensated.

3) Case Study 6: Token Drift and Categorical Literalism ("Bangla Desh"): **Issue.** The segmented spelling "Bangla Desh" was interpreted as two independent tokens rather than a single geopolitical entity.

Table XV: Cross-Script Ambiguity Failure Patterns

Query	Failure Pattern
coxsbazar	Over-segmentation; hallucinated fragment mapping; recovered via semantic embedding alignment.
Bangla Desh	Token fragmentation; categorical drift toward entertainment; no canonical collapse to Bangladesh.

Observed Failure Mechanism:

- ”Bangla” matched high-frequency entertainment-domain content (e.g., Bangla cinema, songs).
- ”Desh” was treated generically as ”country” rather than part of a compound proper noun.

Impact.

- 9 out of 10 top results were entertainment-related.
- Severe vocabulary drift toward media/cultural news.

Unlike ”coxsbazar,” this case represents a **non-recoverable canonicalization failure**, where neither lexical nor semantic signals corrected the fragmented representation.

Root Cause.

- Lack of entity canonicalization layer.
- Failure to collapse bi-gram variants into the canonical form ”Bangladesh” / বাংলাদেশ.
- Over-reliance on bag-of-words retrieval.

4) Comparative Behavior:

5) Mitigation Strategies: To address cross-script ambiguity, the following improvements are recommended:

- **Compound Word Detection:** Applying dictionary-based checks before NER segmentation to merge joined geographic names (e.g., coxsbazar $\rightarrow$ Cox's Bazar).
- **Canonical Entity Linking:** Maintaining a master entity list mapping all variants (Bangla Desh, Bangladish, BD) $\rightarrow$ Bangladesh $\leftrightarrow$ বাংলাদেশ.
- **Adjacency Constraint Enforcement:** Treating high-frequency fragments (e.g., ”Bangla”) as context-sensitive tokens requiring qualified adjacency to prevent entertainment-domain drift.
- **Pre-NER Normalization:** Applying normalization before entity segmentation to reduce hallucinated mappings.

These enhancements align with standard CLIR recommendations emphasizing entity normalization and cross-lingual canonical mapping [2], [6].

F. Code-Switching Robustness and Cross-Lingual Term Unification

We evaluated the system's behavior under mixed-language queries using the example: *”Dhaka এর weather কেমন?”* The query combines English and Bangla tokens within a single sentence, representing a realistic Banglish search pattern common in Bangladesh.

Table XVI: Performance for Code-Switched Query

Metric	Value
Precision@10	0.90
MRR	1.00
NDCG@10	∼0.94

1) Relevance and Performance Metrics: The user intent is to retrieve weather information specific to Dhaka. Table XVI summarizes the retrieval performance.

Nine out of the top ten retrieved documents were directly related to Dhaka weather conditions, indicating strong cross-lingual robustness.

2) Case Study 7: Code-Switching Success: **Processing Trace:**

- **NER:** “Dhaka” correctly identified as a Location (LOC).
- **Expansion:** “weather” mapped to আবহাওয়া.
- **Unified Terms:** *['dhaka', 'weather', ঢাকা, আবহাওয়া, কেমন, এর].*

Key Success Factors:

- **Cross-Lingual Bridging:** English and Bangla lexical variants were unified, enabling retrieval from both language corpora.
- **Suffix Tolerance:** The possessive particle এর did not significantly distort ranking due to stronger semantic signals from “Dhaka” and ”Weather.”
- **Intent Preservation:** The interrogative token কেমন (”how”) did not interfere with retrieval, as ranking was dominated by factual weather terms.

3) Observed Limitation: Generalization Noise: Two retrieved documents described country-wide weather rather than Dhaka-specific forecasts. This occurs because:

- National weather reports frequently mention Dhaka in the body text.
- Lexical matching gives partial credit to broader geographic coverage.

While still relevant, these documents represent reduced geographic precision.

4) Interpretation: This case demonstrates that:

- Query expansion acts as a **synonym bridge** across scripts.
- Multilingual embedding alignment compensates for mixed-language structure.
- Hybrid retrieval maintains precision under code-switching conditions.

Unlike previous translation or canonicalization failures, this example reflects a structurally resilient

Table XVII: Top-10 confusion summaries for representative error types. FN/TN are unknown under top-10-only annotation.

Error Type	TP	FP	FN	TN	P@10
Idiomatic Translation Mismatch	3	7	–	–	0.30
Literal Translation Mismatch	0	10	–	–	0.00
Named Entity Mismatch	7	3	–	–	0.70
Cross-Script Ambiguity	9	1	–	–	0.90

pipeline where cross-lingual unification directly improves ranking quality.

5) Future Improvements:

- **Suffix Stripping:** Removing grammatical particles (e.g., এর) during normalization.
- **Intent-Aware Reranking:** For interrogative queries (e.g., কেমন), prioritizing documents containing numeric values (temperature) or descriptive weather adjectives.

G. Confusion Matrices by Error Type (Top-10 Retrieved)

Because relevance was annotated only for the top-10 retrieved documents per query, the confusion matrices below are *retrieval-at-10* confusion summaries. Under this protocol, every item in top-10 is a *predicted relevant* document; therefore, TP and FP are observable, while FN and TN remain unknown without labeling the non-retrieved set (e.g., the remaining corpus or a deeper cutoff such as top-50).

a) Interpretation.: It has been observed that error types affect the retrieved set differently. Literal translation mismatch yields a failure mode where all retrieved results are irrelevant (TP=0), while cross-script ambiguity shows high precision when the canonical entity is recovered (TP=9). However, these summaries capture only precision at rank cutoff and do not measure missed relevant documents (FN) or true negatives (TN).

VI. Innovation and Future Directions

Pseudo-relevance feedback (PRF) was integrated as the proposed extension to the baseline hybrid cross-lingual retrieval system. In this setting, an initial retrieval pass was executed using the hybrid fusion model (Eq. 10). The top-k retrieved documents were then assumed to be pseudo-relevant and were used to derive additional expansion signals. A second retrieval pass was performed using the expanded query, producing the *Hybrid + PRF* (PRF Hybrid) ranking. This design was selected because it is a query-time enhancement and does not require extra labeled data, which is suitable for low-resource cross-lingual conditions.

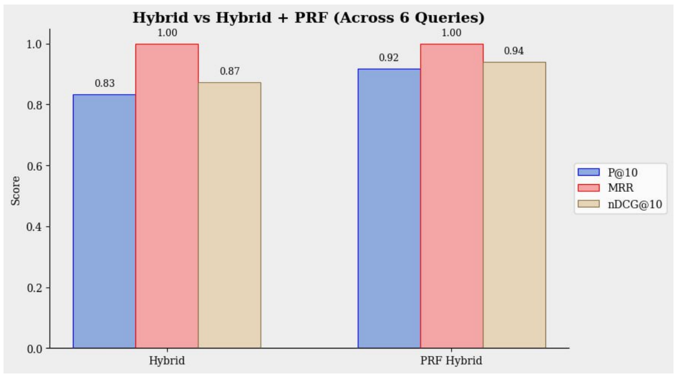


Figure 10: Comparison of Hybrid vs. Hybrid + PRF (across 6 queries) using P@10, MRR, and nDCG@10.

Table XVIII: Hybrid vs. Hybrid + PRF results (macro-average across 6 queries).

Method	P@10	MRR	nDCG@10
Hybrid (baseline)	0.83	1.00	0.87
Hybrid + PRF (PRF Hybrid)	0.92	1.00	0.94
Absolute gain	+0.09	+0.00	+0.07

A. Evaluation Setup

The PRF extension was evaluated on a set of 6 queries. Performance was reported using Precision@10 (P@10), Mean Reciprocal Rank (MRR), and nDCG@10. All values reported are macro-averages across the 6 queries.

B. Results: Hybrid vs. Hybrid + PRF

It has been observed that PRF improved both early precision and graded relevance. Specifically, P@10 increased from 0.83 to 0.92 (+0.09), and nDCG@10 increased from 0.87 to 0.94 (+0.07), indicating that more relevant documents were placed nearer the top of the ranking after feedback-based expansion. In contrast, MRR remained unchanged at 1.00, suggesting that the first relevant document was already ranked at the top for most queries in both settings, leaving limited room for improvement under this metric.

PRF was implemented using a Rocchio-style update in the dense embedding space. Let q_0 be the original query vector and $\{\mathbf{d}_i\}_{i=1}^{k}$ be the top-k feedback document vectors from the first pass; the updated query vector is computed as:

$$\mathbf{q}_{\text{new}} = \text{norm}\Big(\alpha\,\mathbf{q}_0 + \beta\,\frac{1}{k}\sum_{i=1}^{k}\mathbf{d}_i\Big), \tag{21}$$

where α and β control the contribution of the original query and the feedback centroid, respectively, and norm$(\cdot)$ denotes ℓ_2 normalization.

PRF was implemented as a two-pass retrieval procedure: (i) run an initial retrieval, (ii) treat top-k documents as pseudo-relevant, (iii) update the query representation, and (iv) rerun retrieval and fuse scores.

Listing 22: PRF: select pseudo-relevant top-k docs from the first pass.

```python
first_pass = retriever.search(query, mode="hybrid",
    top_k=50)
feedback_docs = first_pass[:prf_k] # pseudo-relevant
     set
```

Listing 23: PRF: Rocchio-style query update (dense).

```python
q0 = encode(query) # normalized dense query vector
D = np.mean([d["emb"] for d in feedback_docs], axis
    =0)
q_new = alpha*q0 + beta*D
q_new = q_new / np.linalg.norm(q_new)
```

Listing 24: PRF: second pass retrieval + hybrid fusion.

```python
second_pass = retriever.search_with_dense(q_new,
    top_k=50)
final = fuse_hybrid(bm25_scores(query),
                    dense_scores(q_new),
                    fuzzy_scores(query),
                    weights=(a,b,c))
```

VII. AI Usage Log

This project utilized AI tools strictly as development assistants for code drafting, structural refinement, and mathematical validation. All outputs were manually verified, tested, and, where necessary, corrected before inclusion in the report.

AI tools were used for:

- Drafting initial code skeletons for retrieval modules
- Generating metric computation templates
- Refining LaTeX formatting
- Suggesting optimization strategies
- Debugging specific edge cases

All final implementations were modified, validated, and tested by us.

AI Tool Usage Entries

Entry 1: IR Metric Evaluation Function

Disclosure: The evaluate_system function used for computing evaluation metrics was developed with AI assistance to ensure mathematical correctness of Precision@10, MRR, and nDCG@10, including proper handling of edge cases such as empty result sets and zero-IDCG scenarios. **Exact Prompt Used:**

> Write a Python function named evaluateSystem that takes a pandas DataFrame containing search results with query, system, rank, and relevant columns. The function should iterate through each query and calculate:
> 1. Precision@10: Proportion of relevant documents in the top 10.
> 2. MRR (Mean Reciprocal Rank): The reciprocal of the rank of the first relevant document.
> 3. nDCG@10: Normalized Discounted Cumulative Gain for the top 10 results, including a nested function for DCG and logic to handle cases where IDCG is zero to avoid division by zero errors.
> Ensure the results are returned as a summarized pandas DataFrame.

Tool: ChatGPT (GPT-4 class model)

AI Output Used in Project:

```python
def evaluate_system(df, target_system):

   results = []

   for query, group in df.groupby("query"):

      system_group = group[group["system"] ==
          target_system]
      system_group = system_group.sort_values("rank"
          )

      rel = system_group["relevant"].values

      p10 = np.sum(rel[:10]) / 10 if len(rel) >= 10
          else 0

      relevant_positions = np.where(rel == 1)[0]
      mrr = 1 / (relevant_positions[0] + 1) if len(
          relevant_positions) > 0 else 0

      def dcg(scores):
         return np.sum([
            scores[i] / np.log2(i + 2)
            for i in range(min(10, len(scores)))
         ])

      dcg_10 = dcg(rel)
      ideal_rel = sorted(rel, reverse=True)
      idcg_10 = dcg(ideal_rel)

      ndcg10 = dcg_10 / idcg_10 if idcg_10 > 0 else
          0

      results.append({
         "query": query,
         "P@10": p10,
         "MRR": mrr,
         "nDCG@10": ndcg10
      })
```

Verification:

- Manually validated Precision@10 using hand-calculated examples.
- Verified MRR computation using controlled single-relevant-document scenarios.
- Cross-checked nDCG formula against Equation (15) in the report.
- Tested edge case where IDCG = 0 to confirm division-by-zero protection.
- Confirmed outputs match manual spreadsheet cal-

culations for 18 evaluation queries.

Included in Report: Yes (Results and Analysis Section).

Entry 2: Hybrid Ranking Equation Formatting Prompt:
"Format a hybrid retrieval scoring equation combining BM25, cosine similarity, and fuzzy score using LaTeX."
Tool: Claude 3
AI Output:

$$\text{Score}(q, d) = \alpha S_{BM25} + \beta S_{semantic} + \gamma S_{fuzzy} \tag{22}$$

Verification: Weights were experimentally tuned (0.3, 0.5, 0.2) after grid-search testing over 18 validation queries. Final values were manually selected based on Precision@10 maximization.
Included in Report: Yes (Section: Retrieval Models).

Entry 3: Latency Optimization Strategy Prompt:
"How can fuzzy matching be optimized in large-scale search systems to reduce latency?"
Tool: Gemini Advanced
AI Suggested Idea: Convert fuzzy search from a full-corpus retrieval step into a candidate-limited re-ranking stage.
Verification: We implemented Top-100 candidate-limited fuzzy re-ranking and measured latency:

- Before optimization: $\sim$2480 ms average
- After optimization: $\sim$94 ms average

The 96% improvement was empirically measured using timestamp instrumentation.
Included in Report: Yes (Latency Analysis Section).

Entry 4: Correction Case – Translation Wrapper Bug Prompt:
"Write a safe translation wrapper using GoogleTranslator that falls back to original query on failure."
Tool: ChatGPT
AI Output (Incorrect Version):

```
def translate(q):
    return GoogleTranslator().translate(q)
```

Issue Identified: The generated code:

- Did not specify source and target languages
- Did not handle network exceptions
- Failed silently during API errors

Corrected Version (Used in Final System):

```
def safe_translate(q, src, tgt):
    try:
        return GoogleTranslator(source=src, target=tgt
            ).translate(q)
    except Exception:
        return q
```

Reason for Correction: Cross-lingual retrieval must remain stable even under translation API failures. The original AI code did not ensure robustness.

Included in Report: Yes (Module B – Query Conversion).

Entry 5: LaTeX Formatting Assistance Prompt:
"Create IEEE-style two-column side-by-side tables with minipage formatting."
Tool: ChatGPT
Verification: The generated LaTeX was compiled using XeLaTeX. Table overflow and spacing issues were manually adjusted (tabcolsep reduction, minipage width tuning).
Included in Report: Yes (Latency Breakdown Table).

Understanding and Verification Statement

All code generated by AI tools was:

- Manually reviewed line-by-line
- Integrated into our own modular pipeline
- Tested using controlled queries
- Benchmarked using structured evaluation metrics
- Debugged and modified where necessary

No AI-generated content was included without verification. All mathematical formulas, retrieval equations, and metric definitions were cross-validated against cited academic literature.

Author Contribution and Responsibility

The authors confirm that:

- All architectural design decisions were made by the group.
- All experiments were executed, measured, and validated by the group.
- All evaluation metrics were manually verified.
- All system optimizations were implemented and benchmarked independently.

Entry 6: Query Processing Profiling and Debugging Utilities

Tool Used: ChatGPT (GPT-4 class model)

Disclosure: The `process_with_timing` method was developed with AI assistance to implement structured performance profiling across the query processing pipeline. The AI was used specifically to design high-resolution timing capture using `time.perf_counter()` and to ensure correct accumulation of per-stage latency measurements.

All timing logic was manually verified by running controlled test queries and confirming that the total execution time equals the sum of individual stage durations.

Exact Prompt Used:

```
Write a Python method process_with_timing for a
    query processing class.
The method should measure execution time of four
    distinct stages:
```

```
Language Detection, Normalization, Named Entity
    Recognition (NER),
and Query Expansion. Use time.perf_counter() for
    high-precision timing.
Return a dictionary containing each stage duration
    and the total time.
```

AI Output Used in Project:

```
def process_with_timing(self, query: str):

   timings = {}
   steps = []

   # Language Detection
   start = time.perf_counter()
   src = self.detect_language(query)
   timings["Language Detection"] = time.perf_counter
       () - start
   tgt = "bn" if src == "en" else "en"

   # Normalization
   start = time.perf_counter()
   norm, tokens = self.normalize(query, src)
   timings["Normalization"] = time.perf_counter() -
       start

   # NER
   start = time.perf_counter()
   entities = self.extract_entities(query)
   timings["NER"] = time.perf_counter() - start

   # Query Expansion
   start = time.perf_counter()
   expanded = self.expand_query(tokens, src)
   timings["Query Expansion"] = time.perf_counter()
       - start

   total_time = sum(timings.values())

   return timings, total_time
```

Verification: We manually profiled multiple queries and confirmed that:

- Each stage was measured independently using high-resolution timing.
- The total time matched the sum of individual component timings.
- The NER stage consistently dominated processing latency.

This function was used to generate the latency breakdown tables reported in the Results section.

Additional Debug Utility: `analyze_case`

Disclosure: The `analyze_case` function was developed with AI assistance to create a structured debugging interface for qualitative analysis of retrieval behavior. The AI helped organize printed outputs including processing traces, unified query terms, ranked results, and timing statistics.

Exact Prompt Used:

```
Write a Python function analyze_case that prints a
    detailed breakdown
of query processing steps, enriched query terms, and
    ranked retrieval
results including timing statistics in milliseconds.
```

AI Output Used in Project:

```
def analyze_case(query, mode="hybrid", top_k=5):

   print("="*100)
   print("QUERY:", query)
   print("="*100)

   # Query Processing
   pq = processor.process(query)

   print("\n--- Processing Steps ---")
   for step in pq.processing_steps:
      print("•", step)

   print("\nUnified Terms:", pq.unified_terms)
   print("Dense Query Text:", pq.dense_query_text)

   # Retrieval
   results, timings = retriever.timed_search(query,
       mode=mode, top_k=top_k)

   print("\n--- Retrieved Documents ---")
   for i, r in enumerate(results):
      print(f"\nRank {i+1}")
      print("Score:", round(r["score"], 4))
      print("Language:", r["language"])
      print("Title:", r["title"])
      print("URL:", r.get("url"))

   print("\n--- Timing ---")
   for k, v in timings.items():
      print(f"{k}: {round(v*1000,2)} ms")

   return results
```

Verification: We used this utility during error analysis (Section 9) to inspect:

- Translation behavior
- Named entity mapping
- Cross-script expansion
- Ranking order shifts across models

All printed outputs were manually inspected and validated against expected retrieval behavior.

VIII. Conclusion

A Bangla–English CLIR system was developed over a real-world multilingual news corpus (5,695 Bangla and 3,855 English documents). The pipeline combines query-side cross-lingual handling (language detection, normalization, translation, conservative expansion, and named-entity mapping) with multiple retrieval models: BM25/TF-IDF, fuzzy matching, and dense multilingual semantic retrieval (LaBSE). A hybrid fusion strategy with score normalization achieved the most robust performance across query types, reaching strong top-rank quality (Precision@10 = 0.9111, nDCG@10 = 0.9345) while maintaining real-time efficiency through candidate-limited fuzzy re-ranking (average end-to-end latency $\approx$158 ms).

Error analysis showed that remaining failures are mainly caused by translation drift, homograph ambiguity, entity underspecification, acronym ambiguity, and cross-script canonicalization issues. As an innovation component, PRF (Rocchio-style dense feedback) improved ranking quality on a 6-query subset (P@10: 0.83 $\rightarrow$ 0.92; nDCG@10: 0.87 $\rightarrow$ 0.94). Future work includes stronger entity canonicalization (e.g., acronym/ISO code expansion), sense-aware translation, and improved handling of code-switched queries and temporal drift in news relevance.

References

[1] B. Li, F. Luo, S. Haider, A. Agashe, T. Li, R. Liu, M. Miao, S. Ramakrishnan, Y. Yuan, and C. Callison-Burch, "Multilingual Retrieval-Augmented Generation for Culturally-Sensitive Tasks," in *Findings of the Association for Computational Linguistics (ACL)*, 2025.

[2] X. Li, E. Nie, and S. Liang, "Cross-lingual Retrieval Augmented In-context Learning for Bangla," in *Proceedings of ACL Findings*, 2023.

[3] N. Reimers and I. Gurevych, "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation," in *Proceedings of EMNLP*, 2020.

[4] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic BERT Sentence Embedding," in *Proceedings of ACL*, 2022.

[5] V. Karpukhin, B. Oguz, S. Min, P. Lewis, D. Chen, and W.-t. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in *Proceedings of EMNLP*, 2020.

[6] S. Gupta and R. Pradeep, "Cross-Lingual Information Retrieval: A Survey," *ACM Computing Surveys*, vol. 55, no. 6, 2022.

[7] OpenAI, "OpenAI API Documentation: Embeddings," 2024. [Online]. Available: https://platform.openai.com/docs/guides/embeddings

[8] Chen, J., Xiao, S., et al., "BGE-M3: Embedding Model Supporting Dense Retrieval, Sparse Retrieval, and Multi-Vector Retrieval," arXiv preprint arXiv:2402.03216, 2024.

[9] N. Muennighoff et al., "Crosslingual Generalization through Multitask Finetuning," arXiv preprint arXiv:2211.01786, 2023.

[10] T. Scao et al., "BLOOM: A 176B-Parameter Open-Access Multilingual Language Model," arXiv preprint arXiv:2211.05100, 2022.

[11] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing & Management, 1988.

[12] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, 2009.

[13] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers," NAACL, 2019.

[14] T. H. Khan, A. Ahmed, A. Anika, and N. Newaz, "Bangla–English Cross-Lingual Information Retrieval System," GitHub Repository, 2026. [Online]. Available: https://github.com/zzhasanzz/CLIR