A Multi-Model Lexical Fusion Approach for English–Bengali Code-Mixed Information Retrieval

Kunal Chakma[1], Subhajit Datta[2]

[1]*National Institute of Technology, Agartala, Tripura, 799046*
[2]*Sister Nivedita University, Newtown, West Bengal, 700156*

Abstract
The CMIR-2025 shared task focuses on retrieving relevant social media posts written in English–Bengali code-mixed text, a setting characterized by noisy transliteration and informal language use. This paper presents a multi-model lexical retrieval framework developed for the task, which integrates two-stage text normalization and hybrid retrieval. The normalization stage combines dictionary-based replacement and fuzzy string matching to handle spelling and transliteration variations. Subsequently, multiple classical models—BM25, TF-IDF, PL2, InL2, and Hiemstra_LM—are applied, and their ranked outputs are merged using Reciprocal Rank Fusion (RRF). The proposed system, submitted under the team name **NITA_CMIR**, achieved a Mean Average Precision (MAP) of 0.1518, nDCG of 0.4151 (ranked **2nd** overall), P@5 of 0.30, and P@10 of 0.237 on the official evaluation dataset. The results highlight the effectiveness of integrating normalization and rank fusion for robust information retrieval in code-mixed social media data.

1. Introduction

Code-mixing (sometimes referred to as code-switching) [1] refers to the linguistic phenomenon where speakers alternate between two or more languages or language varieties within a single conversation, utterance, or even sentence. Words, phrases, or clauses from several languages are included in code-mixed text, which frequently adheres to the grammar of one primary language while including words or structures from another. India is a country of a multilingual society where people (mostly in urban areas) often mix English with other Indian languages during conversations. With the upsurge in social media, users from India often post comments by typing texts in the Roman transliterated form of Indian languages mixed with English. A possible reason for using the Roman transliteration is the standard keypad/keyboard available with the input devices (such as mobile phones and tablets). However, due to the lack of a standardization for Roman transliteration of Indian languages, there are possibilities of spelling variations. For example, the Roman transliterated Bengali word for “me" could be written as “ami", “aami", “amee", or “aamee". There can be inconsistencies in transliteration, and at times, the text may appear noisy because of improper grammatical structures used in social media. Therefore, retrieving relevant texts/documents from social media for a code-mixed query is difficult.

To develop solutions for retrieving English-Bengali code-mixed social media texts, the CMIR-2024 [2] track was organised. Continuing the efforts from the CMIR 2024 shared task, this year’s (CMIR 2025) [3] [4] track aims to advance retrieval methods tailored to noisy multilingual settings, particularly English mixed with Roman transliterated Bengali texts.

In this paper, we describe our NITA_CMIR system, which integrates a two-stage normalization and lexical retrieval with multiple retrieval models. Our main contributions are:

- A two-stage normalization pipeline using dictionary-based replacements and fuzzy matching.
- A retrieval pipeline with multiple classical models and Reciprocal Rank Fusion.
- Competitive performance in CMIR-2025, achieving the 2^{nd} highest nDCG among all the participants.

Forum for Information Retrieval Evaluation, December 17-20, 2025, India
kchakma.cse@faculty.nita.ac.in (K. Chakma); dattasubhajit30@gmail.com (S. Datta)

The remainder of the paper is organized into the following sections. The related works are discussed in Section 2. The dataset is discussed in Section 3. The proposed work is discussed in Section 4. The results are discussed in Section 5, followed by a discussion on the observations in Section 6. The paper concludes in Section 7.

2. Related Works

Code-mixed information retrieval (CMIR) has gained attention in recent years, particularly for low-resource languages such as English-Bengali mixtures common in Indian social networks. Earlier works [5], such as those in the FIRE (Forum for Information Retrieval Evaluation) tracks, focused on handling transliteration and spelling variations in multilingual queries. For instance, the CMIR-2024 shared task [2] introduced benchmarks for retrieving relevant social media posts in English-Bengali code-mixed data, emphasizing the challenges of noisy text and informal language. Chanda and Pal [6] provide an overview of CMIR challenges in FIRE-2024, highlighting issues like orthographic variations and lack of standardization.

Classical lexical models such as BM25 [7] and TF-IDF [8] remain baselines in CMIR due to their efficiency, but they struggle with vocabulary mismatch caused by Romanization and slang [9]. To mitigate this, normalization techniques, including dictionary-based mapping and fuzzy matching, have been proposed [10]. For example, tools such as RapidFuzz[1] have been used to handle orthographic variations in transliterated text. Recent advances incorporate neural embeddings for semantic matching, such as multilingual variants of BERT [11] fine-tuned on code-mixed data [12]. However, these require substantial computational resources, making hybrid approaches, such as combining lexical retrieval with fusion methods such as Reciprocal Rank Fusion (RRF) [13], practical for resource-constrained settings. Our work builds on these by integrating normalization with multiple lexical retrievers and RRF for robust run generation in English-Bengali IR.

3. Dataset

The CMIR-2025 dataset for the shared task, obtained from platforms such as Facebook, comprises approximately 107,900 social media posts in code-mixed English-Bengali text (Romanized Bengali and English). Each post is considered as a document. These documents comprise fields such as DOCNO (unique identifier), HEAD (optional header), and BODY (main content), frequently containing slang, abbreviations, and spelling variations.

The training set provided by the organisers comprises 20 code-mixed queries with the query relevance (qrels). The test set includes 30 code-mixed queries that simulate real-world searches on subjects such as daily life and entertainment. Queries integrate Romanized Bengali (e.g., “ki", “hobe") alongside English.

4. Proposed Work

4.1. Problem definition

The task requires retrieving ranked lists of relevant documents (runs) for English-Bengali code-mixed queries. Code-mixed IR (CMIR) [14], [2], [9] can be defined as retrieving code-mixed documents for given code-mixed queries. Formally, CMIR can be defined as:*for a given query $q \in \langle L^{(i)}, S^{(i)} \rangle$ where $\langle L^{(i)}, S^{(i)} \rangle$ is the language and script, the task is to retrieve the relevant documents from a pool of documents D, written in language $L^{(i)}$ and script $S^{(i)}$, where $i > 1$*. Therefore, in such a scenario, the document pool is defined as:

$$D = \bigcup D_L^{(i)}, S^{(i)} \tag{1}$$

[1]https://github.com/rapidfuzz/RapidFuzz

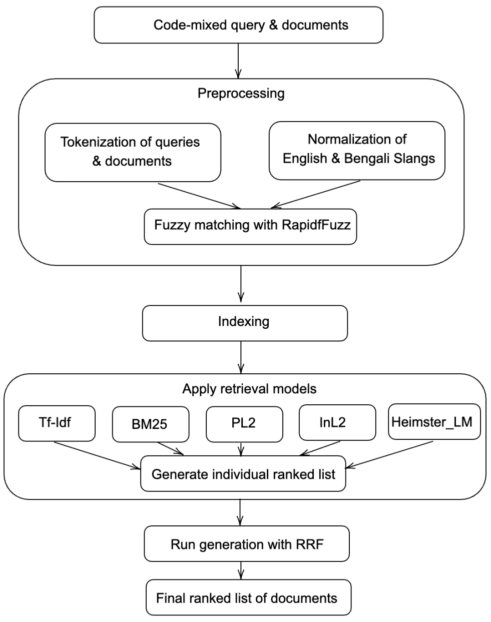
Code-mixed query & documents
Preprocessing
Tokenization of queries & documents
Normalization of English & Bengali Slangs
Fuzzy matching with RapidfFuzz
Indexing
Apply retrieval models
Tf-Idf
BM25
PL2
InL2
Heimster_LM
Generate individual ranked list
Run generation with RRF
Final ranked list of documents

Figure 1: Work flow of NITA_CMIR submitted system.

where $L = \{l_1, l_2, ..., l_i\}$ and script $S = \{s_1, s_2, ..., s_i\}$. In the CMIR 2025 shared task, $L = \{l_1, l_2\}$ and $S = \{s_1\}$.

Our submission uses fused rankings from multiple lexical models. The workflow of our proposed work is shown in figure 1 and described in the following sections.

4.2. Preprocessing Steps

4.2.1. Dictionary normalization

The raw queries and documents contain a significant amount of informal spellings, numeric-letter substitutions, and slang words in both English and Bengali. To address this, we construct normalization dictionaries for both languages.

- **English Slang Dictionary**: Mapping of common abbreviations and non-standard forms such as “gd” $\rightarrow$ “good”, “hpy” $\rightarrow$ “happy”, “frnd” $\rightarrow$ “friend”, “luv” $\rightarrow$ “love”, “gr8” $\rightarrow$ “great”, “plz” $\rightarrow$ “please”, etc., to their standard equivalents. This helps reduce spelling noise in English fragments of code-mixed text.
- **Bengali Slang Dictionary (Romanized)**: Normalization of frequently observed Romanized Bengali slangs. Examples include “ache” $\rightarrow$ “aache”, “6ilo” $\rightarrow$ “chilo”, “onk” $\rightarrow$ “onek”, “erkm” $\rightarrow$ “erokom”, “hye6e” $\rightarrow$ “hoyeche”, and “ki6u” $\rightarrow$ “kichu”.

4.2.2. Tokenization

Following normalization, each sentence is split into tokens. In addition to simple whitespace-based tokenization, we also perform language tagging to identify whether a token belongs to English (en) or Romanized Bengali (bn). Each token is then aligned with its normalized form, if available. Table 1 shows the normalized form of some of the words found in the corpus.

4.2.3. Fuzzy Matching with RapidFuzz

After dictionary-based slang normalization, additional preprocessing is applied to handle spelling variations and noisy Romanized text. We use RapidFuzz, a fuzzy string matching library, to further

Word	Language	Normalized Word
gd	en	good
frnd	en	friend
bdy	en	birthday
6ilo	bn	chilo
krechilo	bn	korechilo
n8	en	night
fvrt	en	favorite
coz	en	because
h66e	bn	hocche

Table 1
Example of Tokenized Representation

normalize tokens. RapidFuzz computes similarity scores (with a threshold of 85) between strings and allows us to map out-of-vocabulary words to their closest standardized form when the similarity exceeds a given threshold.

For example:
"frndz" is mapped to "friend",
"ferends" is mapped to "friends",
"achee" is mapped to "aache".

This two-stage normalization process — first dictionary replacement, then fuzzy matching — reduces the effect of orthographic variations, thereby improving retrieval consistency in code-mixed queries.

4.3. Indexing and Corpus Preparation

After tokenization and normalization, the cleaned corpus is converted into a PyTerrier[2]-compatible format for indexing. The document collection is read into a DataFrame where the `docno` and `title` columns are used directly as the document identifier and text, respectively. All fields are explicitly cast to string types to ensure consistency during indexing.

For the query set, each query is assigned a unique identifier (`qid`) and its text is taken from the `TITLE` column. Queries containing special characters or Bengali script are sanitized to remove characters that could break the Terrier parser. When necessary, queries are translated into English using the Google Translate API[3].

The final corpus is indexed using `IterDictIndexer`, with the document text stored for retrieval and `docno` preserved as metadata. This index serves as the foundation for all subsequent retrieval experiments.

4.4. Retrieval Models

After indexing, we experiment with a set of classical retrieval models provided by `PyTerrier` with the default settings. These models serve as strong baselines for code-mixed information retrieval tasks:

- **TF-IDF** [8]: A vector space model based on term frequency–inverse document frequency weighting.
- **BM25** [7]: A probabilistic retrieval model that normalizes term frequency and document length.
- **PL2** [15]: A divergence-from-randomness model based on the Poisson distribution.
- **InL2** [16]: A variant of divergence-from-randomness that applies the Laplace after-effect.
- **Hiemstra_LM** [17]: A language model-based retrieval approach using linear smoothing.

Each query from the prepared query set passes through these retrieval models. The retrieved results for each query are ranked according to model-specific scores. To leverage the complementary strengths

[2]https://pyterrier.readthedocs.io/en/latest/
[3]https://cloud.google.com/translate

Table 2
CMIR-2025 Results: Team Submissions and Evaluation Metrics on the test data.

Team	Run	MAP	nDCG	P@5	P@10
AiNauts	Run 1	0.043	0.105	0.120	0.080
AiNauts	Run 2	0.035	0.102	0.087	0.077
AiNauts	Run 3	0.054	0.153	0.173	0.130
AiNauts	Run 4	0.009	0.036	0.013	0.060
AiNauts	Run 5	0.027	0.077	0.087	0.067
Defense_NLP	Run 1	0.179	0.366	0.373	0.290
Defense_NLP	Run 2	0.187	0.377	0.393	0.277
NITA_CMIR	Run 1	0.152	0.415	0.300	0.237
MUCS	Run 1	0.082	0.292	0.213	0.157
MUCS	Run 2	**0.212**	**0.486**	**0.420**	**0.300**
MUCS	Run 3	**0.212**	**0.486**	**0.420**	**0.300**
MixMatch IR	Run 1	0.123	0.376	0.293	0.210
MixMatch IR	Run 2	0.111	0.318	0.233	0.167
IRSolver	Run 1	0.064	0.147	0.227	0.157
IRSolver	Run 2	0.057	0.139	0.207	0.157
CodeWeavers	Run 1	0.156	0.341	0.280	0.230
CodeWeavers	Run 2	0.155	0.277	0.380	0.283
NLPFusion	Run 1	0.136	0.319	0.287	0.233
NLPFusion	Run 2	0.136	0.319	0.287	0.233
NLPFusion	Run 3	0.136	0.319	0.287	0.233
NLPFusion	Run 4	0.093	0.247	0.233	0.170
NLPFusion	Run 5	0.093	0.247	0.233	0.170

of all models, the individual retrieval results are later combined using **Reciprocal Rank Fusion (RRF)** [13]. This fusion method produces a single, stronger ranking per query by assigning higher scores to documents that appear near the top across multiple models.

4.5. Run Generation via Reciprocal Rank Fusion

Since *qrels* for the test queries are not provided by the track organizers for the English-Bengali code-mixed dataset, we construct pseudo-qrels using the fused results obtained from Reciprocal Rank Fusion (RRF). The RRF score [13] for a document d for a query is defined as:

$$\text{RRFscore}(d) = \sum_{r \in R} \frac{1}{k + rank_r(d)} \tag{2}$$

where $rank_r(d)$ is the rank of document d in run r, R is the set of all runs, and k is a constant (set to 60). Documents that appear higher across multiple models receive higher RRF scores.

The fused ranking per query is then used to create the pseudo-qrels. Each row contains the `qid`, `docno`, and the RRF-based `score`. These pseudo-qrels serve as a reference for evaluation and further experiments. This approach allows us to leverage the complementary strengths of different retrieval models while keeping the pipeline computationally lightweight.

Since the organizers did not release the official relevance judgments (qrels) for the CMIR-2025 test queries, all experiments beyond the official submissions were performed using pseudo-qrels constructed from the fused RRF rankings. These pseudo-qrels were used only for internal validation, parameter tuning, and ablation analysis on the training data. All leaderboard scores reported in Table 2 correspond to the official evaluation performed by the organizers.

Table 3
Comparison of NITA_CMIR with the top teams. Best scores are in bold; relative gaps for NITA_CMIR are shown in parentheses.

System	nDCG	MAP	P@5	P@10	Leaderboard
MUCS_Run2&3	**0.486**	**0.212**	**0.420**	**0.300**	1st (All scores)
Defense_NLP2	0.377	0.187	0.393	0.290	2nd (MAP)
Defense_NLP1	0.366	0.179	0.367	0.290	3rd (MAP)
NITA_CMIR (Ours)	0.415 (-14.6%)	0.152 (-28.3%)	0.300 (-28.6%)	0.237 (-21.0%)	2nd (nDCG)

5. Results

5.1. Team Performance Analysis

The organizers published the results of the run submissions which are shown in Table 2. These results show that our run achieves an nDCG of 0.415057 (**ranked** 2nd **overall**), a MAP of 0.151773 (**ranked** 5th), a P@5 of 0.300 (**ranked** 7th), and a P@10 of 0.236667 (**ranked** 7th). While our nDCG score is competitive, it does not surpass the top-performing runs (e.g., MUCS runs 2/3, which reach 0.485517). Similarly, our MAP and precision scores trail behind stronger baselines such as MUCS and Defense_NLP, which achieve higher MAP and P@5 values.

These results suggest that, while our retrieval approach is effective in ranking relevant documents highly (as reflected in the nDCG score), it lags in precision-orientated measures. This indicates potential room for improvement in early precision, possibly through better query expansion, reranking, or hybrid ensemble techniques.

5.2. Performance Comparison

A comparison of our submited system (NITA_CMIR) with the top performing systems is shown in Table 3. From Table 3, we observe that our system (NITA_CMIR) is competitive in terms of overall ranking quality, with an nDCG of 0.415, which is only about 14.6% lower than the best-performing system (MUCS_Run2&3). However, our MAP (0.152) is approximately 28.3% below the top score, and our precision at early cutoffs (P@5 and P@10) also lag behind by 28.6% and 21.0%, respectively. These findings highlight that while our approach is effective at ranking relevant documents higher overall, it falls short in early precision compared to the strongest baselines. This indicates the need for improvements in query refinement and reranking strategies to better capture highly relevant documents at the top ranks.

6. Discussion

Our team, **NITA_CMIR**, achieves a strong **nDCG of 0.415**, ranking second overall among all submissions. This indicates that our system is effective at ranking relevant documents higher in the result lists, even though it does not outperform competing runs in MAP or precision-oriented measures (P@5 and P@10).

One possible reason for this performance pattern lies in the nature of nDCG, which emphasizes both *rank position* and *graded relevance*. Our pipeline, which combines dictionary-based normalization with RapidFuzz fuzzy similarity filtering, appears particularly effective at promoting highly relevant documents into the upper ranks. As a result, our system captures the quality of ranking well, though it retrieves fewer relevant documents in the very top positions compared to the strongest systems (e.g., MUCS and Defense_NLP).

The comparatively lower MAP and P@k scores suggest limitations in early precision. This may stem from incomplete query coverage due to limited handling of code-mixing variations, as well as insufficient query expansion and reranking. In addition, reliance on classical retrieval models without deeper neural rerankers could have restricted the system's ability to consistently place the most relevant documents within the top-5 or top-10 results.

Overall, the results demonstrate that our approach is well-suited for ranking effectiveness in noisy, code-mixed social media text, but future work should focus on enhancing early precision through stronger query expansion, entity-aware rewriting, and the integration of neural rerankers.

7. Conclusion

The NITA_CMIR system, developed for the CMIR-2025 shared task on English-Bengali code-mixed retrieval, combines dictionary-based normalization with RapidFuzz fuzzy matching and classical retrieval models in PyTerrier. Despite its lightweight design, the system achieves competitive performance, ranking highly relevant documents near the top. Normalization and fuzzy lexical retrieval are strong bases for code-mixed IR, especially in noisy, low-resource social media contexts. However, performance gaps in MAP and precision suggest the need for enhanced query expansion, entity-aware rewriting, and neural reranking strategies. Future work aims to integrate hybrid approaches to improve ranking quality and early precision.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT and Quillbot in order to: grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] P. Muysken (Ed.), The Cambridge Handbook of Linguistic Code-switching, Cambridge University Press, Cambridge, 2009.

[2] S. Chanda, S. Pal, Overview of the shared task on code-mixed information retrieval from social media data, in: Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation, FIRE '24, Association for Computing Machinery, New York, NY, USA, 2025, p. 29–31. URL: https://doi.org/10.1145/3734947.3735670. doi:10.1145/3734947.3735670.

[3] S. Chanda, K. Tewari, S. Pal, Findings of the code-mixed information retrieval from social media data (cmir) shared task at fire 2025, in: K. Ghosh, T. Mandl, S. Pal, S. Majumdar, A. Chakraborty (Eds.), Forum for Information Retrieval Evaluation (Working Notes) (FIRE 2025) December 17-20, Varanasi, India, CEUR-WS.org, 2025.

[4] S. Chanda, K. Tewari, S. Pal, Overview of the cmir track at fire 2025: Code-mixed information retrieval from social media data, in: FIRE '25: Proceedings of the 17th Annual Meeting of the Forum for Information Retrieval Evaluation. December 17-20, Varanasi, India, Association for Computing Machinery (ACM), New York, NY, USA, 2025.

[5] P. Gupta, K. Bali, R. E. Banchs, M. Choudhury, P. Rosso, Query expansion for mixed-script information retrieval, in: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 677–686. URL: https://doi.org/10.1145/2600428.2609622. doi:10.1145/2600428.2609622.

[6] S. Chanda, S. Pal, The effect of stopword removal on information retrieval for code-mixed data obtained via social media, SN Comput. Sci. 4 (2023). URL: https://doi.org/10.1007/s42979-023-01942-7. doi:10.1007/s42979-023-01942-7.

[7] S. Robertson, H. Zaragoza, The probabilistic relevance framework: Bm25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389. URL: https://doi.org/10.1561/1500000019. doi:10.1561/1500000019.

[8] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing & Management 24 (1988) 513–523. doi:10.1016/0306-4573(88)90021-0.

[9] K. Chakma, A. Das, Cmir: A corpus for evaluation of code mixed information retrieval of hindi-english tweets, Computación y Sistemas 20 (2016) 425–434. URL: https://api.semanticscholar.org/CorpusID:11152913.

[10] T. Mandl, S. Modha, G. K. Shahi, A. K. Jaiswal, D. Nandini, D. Patel, P. Majumder, J. Schäfer, Overview of the HASOC track at FIRE 2020: Hate speech and offensive content identification in indo-european languages, CoRR abs/2108.05927 (2021). URL: https://arxiv.org/abs/2108.05927. arXiv:2108.05927.

[11] T. Pires, E. Schlinger, D. Garrette, How multilingual is multilingual BERT?, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 4996–5001. URL: https://aclanthology.org/P19-1493/. doi:10.18653/v1/P19-1493.

[12] A. Patil, V. Patwardhan, A. Phaltankar, G. Takawane, R. Joshi, Comparative study of pre-trained bert models for code-mixed hindi-english data, in: 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), IEEE, 2023, p. 1–7. URL: http://dx.doi.org/10.1109/I2CT57861.2023.10126273. doi:10.1109/i2ct57861.2023.10126273.

[13] G. V. Cormack, C. L. A. Clarke, S. Buettcher, Reciprocal rank fusion outperforms condorcet and individual rank learning methods, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 758–759. URL: https://doi.org/10.1145/1571941.1572114. doi:10.1145/1571941.1572114.

[14] S. Chanda, S. Pal, Overview of the shared task on code-mixed information retrieval from social media data, in: FIRE 2024 Working Notesl, CEUR Workshop Proceedings, 2024, p. 124–128. URL: https://ceur-ws.org/Vol-4054/T2-1.pdf.

[15] G. Amati, C. J. V. Rijsbergen, Probabilistic models of information retrieval based on measuring the divergence from randomness, ACM Transactions on Information Systems (TOIS) 20 (2002) 357–389. doi:10.1145/582415.582416.

[16] G. Amati, Probabilistic models for information retrieval based on divergence from randomness, Ph.D. thesis, University of Glasgow, 2003. URL: http://theses.gla.ac.uk/1750/.

[17] D. Hiemstra, A linguistically motivated probabilistic model of information retrieval, in: C. Nikolaou, C. Stephanidis (Eds.), Research and Advanced Technology for Digital Libraries, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 569–584.