Chapter 3

Literature Review

In this chapter, we take a tour of a body of existing literature that is relevant to our study goals. We survey a section of the published work in the domain of code-mixed information processing and pinpoint the research gaps, provide context, and narrow down to our research focus. There are a few key areas in the field of code-mixed language processing that need to be investigated. In particular, we revisit the pertinent body of published work in information retrieval, sentiment analysis, word-level language identification, and hate speech identification in this chapter. Each section of the survey is devoted to demonstrating the most recent developments, techniques, and discoveries in these fields.

3.1 Literature Review in Language Identification

Language identification, as a field of study, has a rich historical background that predates the advent of computational methodologies. Initially motivated by the exigencies of translation, early works in this domain relied on rudimentary manual techniques aimed at swiftly identifying the language of documents. Over the course of more than five decades, automatic language identification has garnered sustained scholarly attention, marking significant milestones in its evolution.

Mustonen [28]'s seminal work marked the genesis of formal inquiry into automated language identification. Rau [29] introduced the concept of gapped bigrams, leveraging relative frequencies of character combinations to discern linguistic patterns. Subsequent advancements by Henrich [30] pioneered the application of positive Decision Rules, relying on unique character and character n-grams for language classification. Vitale [31] further expanded the repertoire of techniques, employing character combinations coupled with decision rules to ascertain etymological groupings of proper names, supplemented by trigram analysis for unresolved cases.

The 1990s witnessed a proliferation of methodologies aiming to refine language identification systems. Souter et al. [32] compiled extensive lists of common words for each language, while Giguet [33] meticulously curated exhaustive lists of function words sourced from dictionaries. Kikui [34] delved into the utilization of word suffixes and their relative frequencies for linguistic classification. Giguet's subsequent work in 1998 explored the efficacy of suffix analysis derived from untagged corpora, expanding the scope of language identification techniques.

Mather [35] marked a departure by taking a data-driven approach, collating word frequencies into feature vectors for enhanced categorization. Furthermore, the convergence of language identification with broader text categorization methodologies was underscored by Elworthy [36], laying the groundwork for cross-pollination of techniques between these allied fields. Cavnar and Trenkle [37] proposed a character n-gram approach on a newsgroup article corpus that achieved 99.8% accuracy. However, the same method failed miserably with code-mixed social media data. The reason that most of the newsgroup articles follow formal and standard writing, whereas social media is a collection of informal text can be attributed to this anomaly.

There has been a surge of attention in the computational linguistic field in recent years in supporting code-mixing tasks (Figure 3.1). The first and second workshops on a computational approach to code-switching conducted a shared task on language

identification for numerous language pairs co-located with Empirical Methods in Natural Language Processing (EMNLP) 2014 [1] and EMNLP 2016. Similarly, the Forum for Information in Retrieval Evaluation (FIRE) organized several shared tasks on language identification for Indian language pairs at FIRE 2014, FIRE 2015, and FIRE 2016 [19].

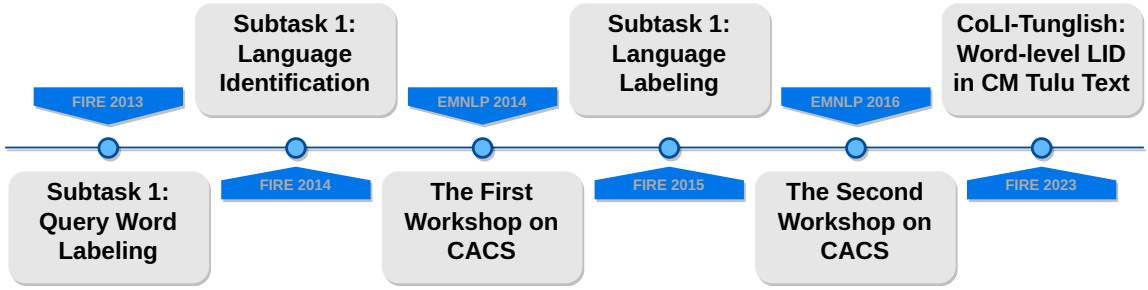

Figure 3.1: List of venues (Shared task) with timeline for Language Identification task

Nguyen and Doğruöz [38] explored different methods like dictionary lookup, language models, logistic regression classifier, and conditional random fields (CRF) classifier for language identification in Turkish-Dutch code-mixed data. For Indian language pairs, the word level language identification has been addressed by Barman et al. [39] at the first workshop on a computational approach to Code-Switching. Different methods like supervised classification (Support Vector Machine (SVM)), Sequence classification (CRF), and dictionary lookup have been applied on Bengali-Hindi-English Facebook comments. Das and Gambäck [40] used the same methods with various features on code-mixed chat message corpora (English-Bengali and English-Hindi). They also introduced a Code Mixed Index (CMI) to evaluate the level of language-blending in a corpus. Vyas et al. [41] created a multi-level annotated corpus of Hindi-English code-mixed text collected from Facebook forums and explored language identification, back-transliteration, normalization, and POS tagging on this data.

In Xia [42], a CRF model was used with features such as word (word vector of

[1] https://emnlp2014.org/workshops/CodeSwitch/call.html

the current term), spellings, and intra-word features. Chittaranjan et al. [43] used a CRF-based system for word-level language identification. As features, word-based and character-based n-gram (3-gram and 5-gram) embeddings are used and passed through a SVM classifier for training and testing [44]. In addition, for word-level language identification, data such as modified edit distance, word frequency, character n-grams, part of speech (POS) tags, and language tags of surrounding words were employed by Jhamtani et al. [45]. Ansari et al. [46] have used POS tags and others features like word length and the word itself to perform language identification in code-switched scenarios. SVM, Decision Trees, Logistic Regression and Random Forests have been experimented with these features.

Many researchers have recently attempted to solve a variety of NLP tasks using different language pairs [47,48]. Some study shows that non-textual features like neighbourhood based features are useful to solve the LID task for more than three languages in a text [49]. For a few language pairs and tasks, there are two centralized benchmarks. Linguistics Code-switching Evaluation (LinCE) [50] is a prominent one that hosts data from four language pairs: Spanish-English [51], Nepali-English [52], Hindi-English [53], and Modern Standard Arabic-Egyptian Arabic. Another forum is GLUECoS [54], a framework for evaluating language understanding in Code-Switched NLP. Language pairings such as English-Hindi and English-Spanish are chosen there.

Although substantial work is done, Language identification still needs to be explored, particularly for the language pairs with scarcity of corpus and related linguistic resources. Within this framework, the CoLI-Kanglish [55] shared task emerges as a significant initiative, spearheaded by organizers committed to open-sourcing a dataset characterized by code-mixed content in Kannada and English, transcribed in Roman script. Lakshmaiah [56] delves into this realm, employing machine learning methodologies to address Code-mixed Language Identification at the word level in Kannada-English texts. Furthermore, the CoLI-Tunglish shared task, held at the Forum for

Information Retrieval Evaluation (FIRE) 2023, extends this exploration to encompass Tulu, a language with its unique linguistic characteristics. This initiative seeks to cultivate learning models tailored for Word-level language identification in Code-mixed Tulu Texts. Notably, the CoLI-Tunglish [57] dataset integrates three languages: Tulu, Kannada, and English at both word and sub-word levels, reflecting the complexity inherent in multilingual communication. Nayek and Joshi [58] introduced L3Cube-HingCorpus, the inaugural large-scale dataset of Hindi-English code-mixed text in Roman script. This corpus encompasses 52.93 million sentences and 1.04 billion tokens, extracted from Twitter. Additionally, they released the L3Cube-HingLID Corpus, the largest dataset for code-mixed Hindi-English language identification (LID), along with HingBERT-LID, a production-quality LID model designed to enhance the capture of code-mixed data as described in their work. The dataset and models are also publicly available [2]. Kalita et al. [59] developed a code-mixed Bodo-English dataset and employed it for word-level language identification tasks using traditional machine learning models. Yasir et al. [60] addressed the issue of mixed script identification within a code-mixed dataset comprising Roman Urdu, Hindi, Saraiki, Bengali, and English. Their language identification model utilizes word vectorization and variants of recurrent neural networks (RNNs). Dutta [61] investigated LID for Bengali-English using a word-level model based on a single bidirectional LSTM with subword embeddings, trained on a very limited code-mixed dataset. Rani et al. [62] presented a new Magahi-Hindi-English (MHE) code-mixed dataset for language identification tasks, where Magahi is a low-resourced language. This corpus enables language identification at both the word and sentence levels.

The effectiveness of [63] for the language identification of Hindi-English code-mixed data has been evaluated using character, sub-word, and word-based representation [64]. This task has been formulated as a token classification task. On top of word repre-

[2] `https://github.com/l3cubepune/code-mixed-nlp`

sentation, convolutional neural network (CNN) and LSTM networks have been used. Jamatia et al. [64] used pre-trained word embedding (GloVe) with LSTM layer and character level Recurrent Neural Network (RNN) with CRF classifier. The authors also demonstrated that when compared to a supervised approach like CRF, the deep learning model achieved comparable accuracy. For text vectorization, the deep learning models used in their preceding two papers used context-independent embedding. For each word/token, this technique is termed prediction-based vectors. The context is lost when all of the various interpretations of a token are combined (averaged) into a single vector. With dynamic language representation, BERT models have recently boosted the language understanding field. On the basis of the words around it, these models construct a contextual representation of the word.

We follow the footsteps of recent works by [63] and [64] on language identification for code-mixed social media data with contextual representation. Non-contextual word representations face limitations in accurately identifying languages. In lieu of employing non-contextual GloVe embeddings, we investigate BERT, a contextual word embedding methodology that has recently demonstrated promising outcomes across various applications.

3.2 Literature Review in Sentiment Analysis

This section summarizes earlier work carried out on sentiment analysis on different code-mixed languages. Sentiment analysis is an exclusive field of study that deals with analysis of people's feelings and views on a given subject. A plethora of studies has been conducted in various languages, focusing on monolingual mono-script setting. However, code-mixed languages throw a different set of challenges due to diverse language-script combinations. Very few code-mixed language pairings have been attempted in till date.

To the best of our knowledge, the first attempt to separate Hindi and English terms from a Hindi-English code-mixed dataset is credited to Sharma et al. [65]. Lexicon

lookup in the corresponding emotion dictionaries was then used to compute sentiment scores. Subsequently, Joshi et al. [66] conducted sentiment analysis on Hindi-English code-mixed data, employing sub-word level representations within an LSTM architecture. This undertaking marked one of the foundational tasks in sentiment analysis on HI-EN code-mixed datasets.

The Sentiment Analysis of Indian Language (Code-Mixed) (SAIL Code-Mixed)[3], a shared task at ICON-2017[4], focused primarily on two widely used code-mixed languages: Hindi-English and Bengali-English. Approximately 13,000 and 2,500 tweets were provided for training and around 5,500 and 3,000 tweets for testing in the Hindi-English and Bengali-English domains, respectively. Notably, nine teams participated in this shared task, with the most successful team using GloVe word embeddings with 300 dimensions alongside TF-IDF scores of word n-grams (ranging from one-gram to tri-grams) and character n-grams (with n varying from 2 to 6). The team employed ensemble voting for classification purposes, comprising three classifiers - linear SVM, logistic regression, random forests, and a linear SVM classifier.

Mandal and Das [67] curated a modest Bengali-English code-mixed dataset based on movie reviews to analyze the role of classifiers and code-mixed factors in Sentiment Identification. Their findings underscored the superior performance of the SVM classifier for both code-mixed training and testing scenarios. Furthermore, SemEval 2020 organized a competition focused on Sentiment Analysis of Code-Mixed Tweets (SentiMix 2020)[5] [68]. The organizers provided approximately 20,000 labeled tweets for Hinglish (Hindi + English) and around 19,000 labeled tweets for Spanglish (Spanish + English). As with previous endeavors, this was a shared task, attracting numerous teams whose submissions showcased the utilization of pre-trained XLM-RoBERTa and mBERT embeddings for their analyses.

[3] http://www.dasdipankar.com/SAILCodeMixed.html
[4] https://ltrc.iiit.ac.in/icon2017/
[5] https://ritual-uh.github.io/sentimix2020/

Even though extensively observed in multilingual countries like India, code-mixing is a global phenomenon. Vilares et al. [69] constructed a synthetic Spanish-English code-mixed dataset for sentiment analysis by amalgamating two monolingual corpora. Their objective was to evaluate the efficacy of supervised models based on the bag-of-words approach, commonly utilized in sentiment analysis tasks. They employed an L2-regularized logistic regression classifier for this purpose. Additionally, an English-Spanish tweet dataset was curated as a benchmark for sentiment analysis [70], with annotations for positive, negative, and neutral sentiments based on SentiStrength. Vilares et al. [71] conducted a comparative analysis of various machine learning methodologies for multilingual polarity classification. Lee [72] proposed a methodology for annotating emotions in a specific corpus of Chinese-English, considering the language(s) in which the emotional expression is articulated. Another dataset pertaining to Chinese code-mixing was compiled from Weibo.com by Wang [73]. Evaluation benchmarks such as GLUECoS [54] and LinCE [74] have been successfully executed, featuring sentiment analysis tasks for Spanish-English code-mixed text.

In the Indian context, although some code-mixed datasets are available involving North Indian languages, more data is needed for experimentation for the same in Dravidian languages. In the last few years, there has been a some interest in Dravidian code-mixed languages. Appidi et al. [75] generated a Kannada-English code-mixed dataset for emotion prediction, leveraging features such as character n-grams, word n-grams, negation words, emoticons, capitalization, and emotion words. They employed SVM and LSTM-based models for classification. Padmaja [76] employed a traditional word probabilities-based method for sentiment analysis on movies, from Telugu-English tweet collection. Kusampudi et al. [77] introduced an annotated dataset for Sentiment Analysis on Code-Mixed Telugu-English Text (CMTET) alongside a novel unsupervised data normalization method utilizing a Multilayer Perceptron (MLP) model.

Special tracks named Dravidian CodeMix sentiment analysis were organized by

FIRE in 2020 and 2021, featuring datasets comprising a limited number of sentences from YouTube comments in code-mixed Kannada-English, Malayalam-English, and Tamil-English. Advanced techniques like TF-IDF, FastText [78], Transformer-based models, multilingual word embeddings, transfer learning, and meta-embeddings [79], were employed to discern sentiment in Dravidian language pairs. Notably, mBERT [6] and XLM-Roberta-based models [7] achieved state-of-the-art performance. The primary objective of this shared task was to ascertain the sentiment polarity of code-mixed data originating from Dravidian language pairs, specifically Malayalam-English, Tamil-English, and Kannada-English, extracted from the comment sections of YouTube videos sourced from social media platforms. Categorically, the text was classified into five distinct categories: Positive, Negative, mixed feelings, unknown state, and not-language. This endeavor has brought forth several pertinent research issues, with the datasets facilitating further exploration and investigation. Of particular note are the challenges associated with language identification and sentiment analysis, each presenting unique complexities that current state-of-the-art systems are yet to adequately address. Notably, there remains a lack of consensus regarding the optimal strategies for conducting sentiment analysis when faced with datasets comprising a diverse array of textual compositions, encompassing pure monolingual, transliterated monolingual, single-script multilingual, and mixed-script multilingual expressions.

3.3 Literature Review in Hate speech and Offensive Content Identification

The identification of hate speech and offensive content has attracted considerable attention within both academic and commercial spheres. Although a substantial volume of research has predominantly focused on English, owing to its widespread global us-

[6]https://huggingface.co/bert-base-multilingual-cased
[7]https://huggingface.co/xlm-roberta-base

age, there exists an urgent requirement for equivalent corpora in other languages to comprehensively tackle this issue. Several studies have delved into the varied aspects of offensive content, such as *abusive language* [80, 81], *cyber-aggression* [82], *cyber-bullying* [83, 84], and *toxic comments or hate speech* [85–87]. A brief overview of some notable works in these areas is provided.

- *Hate Speech Identification:* Hate speech, a pervasive challenge, has been systematically categorized into various types based on the nature of its textual content. Diverse datasets have been curated to cater to these distinct categories of hate speech. Notably, a common dataset [88] has served as a foundation for identifying hate speech and profanity, with a recent work by Davidson et al. [86] making use of a dataset comprising nearly 24,000 labeled tweets.
- *Offensive Content and Cyberbullying:* The broader domain of offensive content encompasses use of abusive languages [82], cyber-aggression, cyber-bullying, and toxic comments. Previous investigations have employed techniques such as sentiment analysis, topic modeling [83], and user-related features [84] to tackle this multifaceted problem.

Efforts have extended beyond English, with endeavors in languages including German [89,90], Spanish, Arabic [81,91], Greek [92], Slovene [93] and Chinese [94]. Mubarak et al. [81] introduced a collection of profane terms, known as SeedWords (SW), and applied the Log Odds Ratio (LOR) to individual word unigrams and bigrams. Saroj et al. [95] adopted a Support Vector Machine (SVM) approach alongside TF-IDF features, targeting hate speech and offensive language in Arabic and Greek.

In recent years, initiatives like HASOC [89] and GermEval [96] have spotlighted the importance of addressing hate speech detection in various languages and contexts. Dravidian LangTech [97], for example, focused on detecting offensive language in a code-mixed dataset comprising Tamil–English, Malayalam–English, and Kannada–English. The application of multilingual models, including BERT variants and IndicBERT, has

shown promise in this regard. Transfer learning has also shown potential in enhancing offensive language recognition, particularly in code-mixed contexts. Researchers have leveraged transfer learning from English datasets to improve offensive language recognition in code-mixed Kannada [98], Malayalam [99], and Tamil [100].

Detecting hate speech within conversational Hindi-English code-mixed data introduces additional complexities owing to the unspoken and implied conversational context in such content. The hierarchical structure comprising posts, comments, and replies mandates a nuanced approach, necessitating a spectrum of techniques ranging from unified text treatment to innovative hierarchical neural network architectures. Each post may engender multiple comments, and each comment may elicit several replies. Within the domain of English-Hindi data, every component of the tuple can exhibit code-mixing between Hindi and English, solely English expression, exclusively Hindi communication, Romanized Hindi representation, or a fusion thereof. Consequently, intricate input patterns manifest. The assignment of labels to replies or comments is notably influenced by the contextual information imparted by the parent text. Bagora et al., [101] proposed a novel hierarchical neural network architecture, while Madhu et al., [102] employed a pipeline consisting of an LSTM classifier followed by a fine-tuned SentBERT model for hate speech identification. HASOC Hindi-English code-mixed dataset serves as the foundation for our investigation into conversational hate speech identification.

Majority of the above state-of-the-art models work on the single standalone sentences, but do not take into account the nuanced relation among the sentences or the context spanned across hierarchies of several replies and mentions which is required in contextual hate speech recognition. Within this framework, our approach necessitates initial coarse-grained classification to distinguish between instances of hate and non-hate speech. Subsequently, fine-grained classification is required to delineate the specific nature of hate speech, namely, whether it manifests as single, isolated instances

or within a contextual framework. This nuanced task emerges as the most formidable challenge for current models.

3.4 Literature Review in Code-Mixed Information Retrieval

In a multilingual country like India, it is commonplace among social media users to write about their information needs in the form of queries in different languages (mostly in their native languages) but using Roman script. Although a number of works focused on monolingual IR activities exist across all languages, only a few studies are done on Cross-lingual Information Retrieval (CLIR) and Mixed-Script Information Retrieval (MSIR). Ganguly et al. [103] studied code-mixed and monolingual indexing and retrieval algorithms using a code-mixed Twitter collection containing Bengali, Hindi and English tweets. In their tests, they used unigram and bigram from the collection vocabulary to generate a query set. They made certain that no Bengali or Hindi was used in queries, but wanted to see how frequently the code-mixed documents can be retrieved at the top for a monolingual query. They investigated three separate indexing strategies (language-separate monolingual indexing, code-mixed indexing and cluster of the two) to see the effect on code-mixed information retrieval.

Microsoft Research India (MSRI) introduced Mixed Script Information Retrieval (MSIR) track at the Forum of Information Retrieval Evaluation (FIRE) Conference in 2013 and ran several tracks in subsequent years till 2016. Different tasks under MSIR can be classified into the following categories.

- Transliterated Search
- Ad hoc retrieval for Hindi Song Lyrics
- Code-mixed Cross-script Question Answering
- IR on Code-Mixed Hindi-English Tweets

The MSIR track at FIRE 2013 was divided into two sub-tasks as follows.

1. **Query Word Labelling:** Here queries were in Roman script but in Hindi and

English, and the task was to put a language label on each query term and back-transliterate the Hindi words.

2. **Hindi song lyrics retrieval** was based on queries where both documents and queries were in either Roman or Devanagari or mixed script (both Roman and Devanagari).

The sub-task of song lyrics retrieval was organized for three consecutive years, from 2013 to 2015. In 2013 the task was 'Multi-script Ad hoc retrieval for Hindi Song Lyrics', where the query was in Roman script, and documents were in mixed script (Roman and Devanagari). In the year 2014, the task was renamed as 'Mixed-Script Ad hoc Retrieval for Hindi Song Lyrics', where the query and documents can be in either Roman or Devanagari script, but not mixed. In addition to that, the Roman queries could be composed of compound words that might mean something completely different when split. In 2015 the queries were in Roman or Devanagari, and the documents were in mixed script. For the first two years, the queries and documents were Hindi song lyrics (HSL). However, for the year 2015, the queries and documents were in different genres: Hindi song lyrics (HSL), movie reviews (MR), and astrology. The details of the three years task description are mentioned below.

Year	Query	Documents	Data
2013	Roman	Mixed(Roman& Devanagari)	HSL
2014	Roman/Devanagari	Roman/Devanagari	HSL
2015	Roman/Devanagari	Mixed(Roman & Devanagari)	HSL, MR, astrology

In 2013, for the 'Multi-script Ad hoc retrieval for Hindi Song Lyrics' subtask only three teams submitted their runs. The descriptions and results are mentioned in Table 3.1. In 2014 for the same subtask four teams submitted seven runs in total. All the descriptions and results for these runs are mentioned in Table 3.2. In 2015 total 12 runs were submitted by four teams (Table 3.3).

At FIRE 2016, the subtask 'Information Retrieval on Code-Mixed Hindi-English Tweets' was introduced, where the queries and documents both were code-mixed. The

Table 3.1: System papers of MSIR 2013 task: Ad hoc retrieval for Hindi song lyrics

Team Name	Run	Methodology	MAP
TUVal [104]	1	The song collection was indexed as word bi-grams. They employed an autoencoder for intra/inter script spelling variances. Different parameter values used: min_len = 2, θ = 0.95, model = tf_idf where, min_len is the minimum term length for term-matching and θ is the similarity threshold.	0.421
	2	min_len = 2, θ = 0.95, model = XSrqA_M	0.424
	3	min_len = 3, θ = 0.95, model = XSrqA_M	0.356
GU [105]	1	Queries in Roman transliterated and both Roman and Devanagari forms are concatenated, model: tf_idf	0.255
	2	The queries were retained in Roman only, model: tf_idf.	0.216
NTNU [106]	1	Lucene was used to index the documents. The query was retained in Roman script only. model: tf_idf.	0.003
	2	The entire query was transliterated to the Devanagari script.	0.152
	3	Hindi words written in Roman were back transliterated to Devanagari script, but English words were retained in Roman.	0.197

query terms were mostly named entities, and the average length of the queries was 3.5 words. For 'Information Retrieval on Code-Mixed Hindi-English Tweets' subtask, approaches and reported official scores of different participants are shown in Table 3.4. A few teams submitted runs, but did not disclose their approaches.

Chakma and Das [10] attempted to address the challenges, for the first time, connected to information retrieval with Code-Mixed Indian social media texts, mainly tweets. They extracted Hindi-English tweets on topics related to the trending events. The authors began by collecting entities related to the topics of *Delhi Assembly Election* that took place and produced many such election-related posts, political campaigning, and results. Bag of words approach was followed for both Hindi and English words with Hindi words in Roman transliterated form. Such terms were combined with the named entities and queries were generated. For example, the query was *BJP aur Kejriwal* regarding the Delhi election where *BJP* and *Kejriwal* were used as a named entity and *aur* (and) as a Hindi term. Another query was *aam aadmi party Delhi mein*, where

Table 3.2: System papers of MSIR 2014 task: Ad hoc retrieval for Hindi song lyrics

Team Name	**Run**	**Methodology**	**MAP**
BIT [107]	1	word tri-gram query expansion	0.2698
	2	word tri-grams + most common word-grams. model: tf_idf	0.3414
BITS-L [108]	1	Using the Google transliteration, the queries and documents were back-transliterated to Devanagari. Vowels were eliminated and character n-grams were indexed as part of the normalisation process (n ranged from 3 to 6).	0.6263
	2	On top of the above, consonant mappings tried	0.6421
DCU [109]	1	Character sequences of transliterated words normalized using a rule-based approach. A dictionary was used to extend documents and queries. To accommodate for morphological variances of the transliterated words, prefix-matched fuzzy query phrases were employed during retrieval.	0.4112
	2	Keeping all settings the same as in Run 1, phrase-based statistical machine translation (PBSMT) was used in addition.	0.2063
IIITH [110]	1	Using a transliteration approach, all documents and queries were transformed into Roman script. To achieve a better representation of a word, some normalization was used. Edit distance was used to address term variation in the Roman script.	0.412

aam aadmi party and *Delhi* were named entities, and *mein* (in) was a Hindi term. The majority of the Hindi words (*aur*, *mein*) were actually stop words. Other popular topics, such as the Cricket World Cup 2015 and the Bollywood film actors controversies, were collected in a similar fashion. The author applied the Boolean model on 20 queries with an average query length of 2.67 terms. A MAP value of 0.186 was achieved.

Initiatives in code-mixed IR has so far been very few, primarily due to the formidable challenge associated with dataset acquisition. The absence of publicly available datasets specifically tailored for code-mixed information retrieval tasks presents a significant hurdle. Furthermore, while existing transliteration search frameworks primarily cater to single-language contexts and code-mixed scenarios necessitate the handling of multiple languages concurrently, a dedicated effort is needed to build the dataset and then to carry out experiments in this growing field.

Table 3.3: System papers of MSIR 2015 task: Ad hoc retrieval for Hindi song lyrics

Team Name	Run	Methodology	MAP
AmritaCEN	1	did not reveal their strategies.	0.0986
BIT-M [111]	1	The system consisted of two components. The transliteration module converted words from Roman script to Devanagari script. Devanagari script was then used to perform search on a bi-gram representation of the documents.	0.3922
Watchdogs [112]	1	Lucene was used to construct an index after transliterating all documents and queries from Roman script to Devanagari script.	0.3173
	2	After transliteration, all white spaces and vowel signs were discarded, and character-level n-grams were indexed (where n range from 2 to 6).	0.2922
	3	All white spaces were removed after transliteration, and letters used for vowel sounds were substituted with the actual vowels. The documents were then indexed using character-level n-grams (with n ranging from 2 to 6).	0.3814
	4	Following transliteration, the documents were indexed using a word-level n-gram (where n ranged from 2 to 6).	0.2360
ISMD [113]	1	With the original query, the author employed basic indexing.	0.0928
	2	Indexing was as above, but queries were intelligently re-formulated. A transliterated query was combined with the original query for searching.	0.1444
	3	With the original query, the author used block indexing (with a block size of two words).	0.0954
	4	Used block indexing with a re-formulated query.	0.1507
QAIIITH	1	The team did not reveal their strategies.	0.0705
	2	The team did not reveal their strategies.	0.0561

Table 3.4: Approaches on 'IR on code-mixed Hindi–English tweets', FIRE 2016

Team Name	Run	Model	MAP
UB [114]	1	They employed Named Entity boosts, which provided a numeric boost to the documents when the query yielded a NE match. They also ranked documents where the phrase matched the input query.	0.0217
	2	The query was expanded with synonyms using Edit distance, and the rest of the ranking procedure was same as in run 1.	0.016
	3	The author applied NEs as a narrative-based weighting in this experiment.	0.0152
Anuj	1	The approach used by the team was not disclosed.	0.0209
	2	The approach used by the team was not disclosed.	0.0199
Amrita_CEN [115]	1	They deleted several stop words in Hindi and English and tokenized all of the tweets. Following the development of the Term Document Matrix, the Non-Negative Matrix Factorization was applied to queries and tweets to get the distributional representation. Finally, the cosine distance was computed to identify the top relevant tweets.	0.0377
NLP-NITMZ	1	The team did not provide their working notes.	0.0203
NITA_NITMZ	1	The team did not provide their working notes.	0.0047
CEN@Amrita [116]	1	They computed the similarity between the tweets and the given query using Vector Space Models.	0.0315
	2	In this run, they manually eliminated certain Hindi stop words (such as ka, jo, ke ne, to, ve, le), and the rest of the procedures were the same as in run 1.	0.0168
IIT(ISM)D	1	According to the organizer's overview paper [117], the team treated each tweet as a document and indexed it using uniword indexing. The Soundex coding system was used to extend query terms. Three retrieval models (BM25, DFR, and TF-IDF) were used. This team, however, submitted the runs after the deadline.	0.0021
	2	The team did not provide their working notes.	0.0083
	3	The team did not provide their working notes.	0.0021