Information Retrieval for Code Mixed Indian social media text

Report submitted in fulfillment of the requirements for the B.Tech. Project of

Fourth Year B.Tech.

by

S Ujjwal
18075052
Shayak Das
18075056
Ayush Damele
18075014

Under the guidance of
Dr Sukomal Pal

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
December 2021

Dedicated to

our parents, supervisor and professor for their constant guidance and support in accomplishing the task.

Declaration

I certify that

1. The work contained in this report is original and has been done by ourselves and the general supervision of our supervisor.

2. The work has not been submitted for any project.

3. Whenever we have used materials (data, theoretical analysis, results) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever we have quoted written materials from other sources, we have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi

S Ujjwal
Shayak Das
Ayush Damele

Date: 5/Dec/2021

B.Tech. Student
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Information Retrieval for Code Mixed Indian social media text**" being submitted by **S Ujjwal (Roll No. 18075052)**, **Shayak Das (Roll No. 18075056)**, **Ayush Damele (Roll No. 18075014)** , carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date: 5/Dec/2021

Dr Sukomal Pal
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to extend our gratitude to Prof. Sukomal Pal to give us the opportunity to invest in this B.Tech. Project. The help and effort given by our mentor Mr Supriya Chanda proved to be indispensable in the completion of this project.

Place: IIT (BHU) Varanasi

Date: 5/Dec/2020

S Ujjwal

Shayak Das

Ayush Damele

Abstract

In the last decade, natural language processing (NLP) techniques have become commonplace. The processing of a single language is responsible for the majority of these advancements. With the widespread adoption of social media platforms, the focus has shifted to code-mixed text. Text written in multiple languages makes up the code-mixed text. People naturally combine their native tongue with global languages such as English. Current NLP techniques are insufficient to process such texts. As a result, information retrieval (IR) on such Indian social media data is a difficult problem when the documents and queries are a mixture of two or more languages written in either native scripts or Roman transliterated form.

The text is first processed to determine the language of the words in the text. We focus on language identification in Bengali-English and Hindi-English code-mixed sentences in this paper. We then focused on Information Retrieval (IR) difficulties for Code-Mixed Indian social media communications, namely texts from Facebook, Telegram, and Twitter. We documented limits of current state-of-the-art IR systems on such data using our corpus collection and formalised the problem of Code-Mixed Information Retrieval on Informal Texts through query expansion using a hybrid technique based on phonetic matching algorithms. We also explored the usage of Neural Information Retrieval (Zero Shot Learning) on code-mixed data to analyze the performance of different pre-trained models when used as re-rankers. We also performed Named Entity Recognition using Multilingual Meta Embeddings.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Web 2.0 has resulted in an exponential increase in the number of Web users and the volume of Web content. The majority of these users are both consumers and producers of information. People use Roman script to express themselves in colloquial languages (transliteration). Because these texts are mostly informal and casual, grammar rules are rarely followed. In transliterated text, there is no established set of spelling rules. This liberation leads to large-scale spelling variations, which pose a significant challenge in the processing of mixed script data [1]. Recently, with the widespread adoption of social media platforms, the focus has shifted to code-mixed text. Text written in multiple languages makes up the code-mixed text. People naturally combine their native tongue with global languages such as English. Current NLP techniques are insufficient to process such texts. The text is first processed to determine the language of the words in the text [2]. The coarse nature of code-mixed social media text makes language identification challenging. Then we use query expansion techniques using phonetic matching algorithms before retrieving data from document corpus. We also explore the method of zero shot learning for code mixed data.

1.2 Motivation of the Research Work

It's crucial to remember that humans are capable of creating new language registers and learning new communication methods. Not only are we good at it with human-human communication, but we can also build and learn to use efficient registers for human-machine communication, such as Linux command-line expressions or Alexa interaction grammar. We must be able to understand what is being said in these various registers if we want machines to participate in such human-human conversations.

Understanding code-switched communication will help large corporations better target their advertising. Understanding genuine user feedback on product features aids in the development of future versions. Ignoring one language in favour of another or completely ignoring code-switched languages can lead to incorrect conclusions about user sentiment. Understanding how people feel, if they are being open, will help healthcare providers provide better care, improve communication with patients, and improve the distribution and uptake of preventative care. Communication in the appropriate register for tutoring, as well as an understanding of whether or not concepts have been grasped, is critical for educators. For entertainment purposes, characters that do not play should contact and/or understand natural, code-switched communication with other players.

Chapter 2

Code Mixing

2.1 Overview

Code-mixing is a phenomenon that happens when people speak in more than one language at the same time, or switch between two languages to the point where they transition from one tongue to the other in a single sentence. Without changing the topic, code mixing can occur at multiple levels of language, including phonology, morphology, grammatical structures, and lexical items. We can not ignore the fact that the first language has a significant impact on the second. Languages mix and interact, resulting in a variety of languages. The majority of people in society blend their language with other languages through borrowing or utilizing fragments of other languages, while they are occasionally still impacted by their initial language.

Code-mixing is a type of linguistic choice that is more delicate than code-switching. Pieces of one language are utilised in code-mixed phrases, yet the speaker is speaking in a different language. Words are the most common 'pieces' of the other language, but phrases and bigger units can also be used. Thus, we can note that the notion of code-mixing is confined to code shifts inside the same phrase or sentence.

Different types of Code-Mixing in social media data[4]:

- **Inter-Sentential**: the language switch is done at sentence boundaries. This type is seen most often in fluent bilingual speakers.

  Example:

  Will college reopen? Karor kache khobor ache kichu?

  English translation: Will college reopen? Does anyone have any news?

- **Intra-Sentential**: the shift is done in the middle of a sentence, with no interruptions, hesitations, or pauses to indicate a shift. The speaker is usually unaware of the shift.

  Example:

  Porer semester e college kholar kono chance ache ki?

  English translation: Is there any chance that college will reopen next semester?

- **Intra-word**: It occurs within a word itself.

  Example:

  Amar schooler kotha khub mone pore at times.

  English translation: I miss school at times.

2.2 Code-Mixing Metrics

To know the level of code-mixing between languages, we compute three different metrics for the datasets.

1. **Multilingual Index (M-Index)**: The Multilingual Index (M-Index) [3] determines how many languages are utilised in a written document having at least two languages. The index ranges from 0 (monolingual corpus) to 1 (terms in a specific text are evenly shared across languages or each language is represented by an equal number of tokens). The M-Index is calculated as follows:

$$\text{M-Index} = \frac{1 - \sum_{j=2}^{N} p_j^2}{(N-1)\sum_{j=2}^{N} p_j^2} \tag{2.1}$$

Where $N > 1$ is the total number of languages present in the corpus and p_j is the probability of presence for language j in the corpus.

The M-Index, on the other hand, does not reveal the degree of cohesiveness among the languages (i.e., how the languages are dispersed inside a phrase).

2. **Integration Index (I-Index)**: Integration Index (I-Index) [4] is a metric that includes the probability of switching within a text. Given a corpus composed of tokens tagged by language l_i where i ranges from 1 to $v-1$, and j ranges from $i+1$ to v (considering a pair of words in a sliding window of two words in a sentence). The I-index is calculated as follows,

$$\text{I-Index} = \frac{1}{v-1} \sum_{1 \leq i < j \leq v} S(l_i, l_j) \tag{2.2}$$

where switch point, $S(l_i, l_j) = 1$ if $l_i \neq l_j$ and 0 otherwise. n is the total number of tokens present in the corpus, u is the number of tokens given *other* tags. v is the number of tokens given language tags ($v = n - u$). The factor of $1/(v—1)$ reflects the fact that there are maximum $(v—1)$ possible switch points in a corpus of size n.

The I-index is an intuitive measure of how many Code-Switches (CS) are there in a document. Here, value 0 represents a monolingual text with no switching and 1 represents maximum code-switches, i.e. a text in which every word switches language, an improbable real-world situation.

3. **Code-Mixing Index (CMI)**: It is desirable to have a measurement of the level of mixing between languages when comparing different code-mixed corpora. To this end, we present CMI[5], a code-mixing index. This is done at the

utterance level by identifying the most common language in the utterance and then counting the frequency of words from all other languages present.

$$CMI = \frac{\sum_{i=1}^{N}(w_i) - max\{w_i\}}{n - u} \tag{2.3}$$

where $\sum_{1}^{N}(w_i)$ is the sum over all N languages present in the utterance of their respective number of words, $max\{w_i\}$ is the highest number of words present from any language (regardless of if more than one language has the same highest word count), n is the total number of tokens, and u is the number of tokens given language independent tags.

CMI-all is an average over all the utterances in a corpus, and **CMI-mixed** is an average over only code-mixed utterances in a corpus.

Chapter 3

Language Identification

3.1 Overview

The main aim of this work is to compare context free (Word2Vec, GloVe, FastText) and contextual (BERT) input representation in Code-Mixed data automated language recognition. This article focuses on three language pairs: English-Hindi, English-Bengali, and Spanish-English. These language combinations were chosen because Hindi (with approximately 342 million native speakers in northern India and parts of Pakistan), Bengali (with approximately 229 million native speakers in eastern India and Bangladesh), and Spanish (with approximately 500 million native speakers worldwide) are the world's fourth, seventh, and second most spoken languages. Language Identification can be used in the task of Information retrieval as it can allow us to know the degree of code mixing and also the languages involved in a query, thus allowing the usage of different query expansion strategies depending on the languages.

3.2 Related works

There has been a lot of interest from the computational linguistic community to support code- mixing tasks in the recent past. The first and second workshop on a computational approach to code-switching conducted a shared task co-located with EMNLP 2014 a and EMNLP 2016 on lan- guage identification for many language pairs. In addition, 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.

The effectiveness of Joshi et al. (2020) for the task of Hindi-English language identification has been evaluated using character, sub-word, and word-based representation. This task has been formulated as a token classification task. On top of word representation, the most popular convo- lutional neural network (CNN) and long short term memory (LSTM) networks have been used. Jamatia et al. (2019) used pre-trained word embedding (GloVe) with LSTM layer and Character level Recurrent Neural Network (RNN) with Conditional Random Fields (CRF) classifier.

We follow the footsteps of recent works by Joshi et al. (2020) and Jamatia et al. (2019) on language identification for code-mixed social media data with contextual representation. However, instead of using context-free GloVe, we explore BERT, a contextual word embedding that have recently shown to produce good results in other applications.

3.3 Dataset

In this project, we worked on three code-mixed corpora, one of them were collected from the popular social media platforms Facebook, Twitter and WhatsApp and the other two are *SAIL(Bengali-English) Dataset* and *SAIL(Hindi-English) Dataset*[2]. The data sets also has each and every word tagged with its corresponding language. The sentiment tags are ignored and only language information is used for the experiments. The code mixed text in this data set was extracted from twitter and manually annotated for sentiment and language.

SAIL(Bengali-English) Dataset			
	Train	Dev	Test
Total Sentences	3038	2500	1166
Total tokens	57410	44295	21834
Monolingual	481	188	191
Codemixed	2557	2312	975

Table 3.1: SAIL(Bengali-English) Dataset

SAIL(Hindi-English) Dataset			
	Train	Dev	Test
Total Sentences	12482	2968	3010
Total tokens	193702	46156	46116
Monolingual	5449	1307	1286
Codemixed	7033	1661	1723

Table 3.2: SAIL(Hindi-English) Dataset

ICON POS Dataset(Bengali-English)			
	Train	Dev	Test
Total Sentences	1852	618	618
Total tokens	20998	7505	7200
Monolingual	1344	440	454
Codemixed	508	178	164

Table 3.3: ICON POS Dataset(Bengali-English)

ICON POS Dataset(Hindi-English)			
	Train	Dev	Test
Total Sentences	1578	526	527
Total tokens	23920	8747	8475
Monolingual	640	217	228
Codemixed	938	309	299

Table 3.4: ICON POS Dataset(Hindi-English)

3.4 Language Identification

3.4.1 Model Architecture

The model architecture basically comprises of three layers:

LinCE Dataset(Hindi-English)			
	Train	Dev	Test
Total Sentences	4823	744	1854
Total tokens	95224	15446	36052
Monolingual	2702	419	-
Codemixed	2121	325	-

Table 3.5: LinCE Dataset(Hindi-English)

LinCE Dataset(Spanish-English)			
	Train	Dev	Test
Total Sentences	21030	3332	8289
Total tokens	253221	40391	97341
Monolingual	12851	2182	-
Codemixed	8179	1150	-

Table 3.6: LinCE Dataset(Spanish-English)

- **Embedding Layer**: Word Embeddings are dense representations of individual words in a text that take into account the meaning and other words with which the individual word appears.This layer converts each word in the input sequence into a vector which is an essential step as the model needs to take input in the form of vectors. Different embedding models can be used in this layer. In our case we used:

  1. Word2Vec
  2. FastText [6]
  3. Glove
  4. BERT [7](Best performance)

- **Word Sequence Layer Layer**: This layer is used to generate a word sequence representation. This layer takes the sequence of embedding vectors of the text sequence as input and captures the context of the surrounding words to generate new vectors. Different algorithms used in this layer are:

1. LSTM

2. Variants of CNN

- **Inference Layer**: This final layer is made up of softmax feedforward networks which generates the probability distribution corresponding to each word of the sequence over the list of all the tags. For each word we take the tag with the highest probability as the predicted tag during the prediction stage.

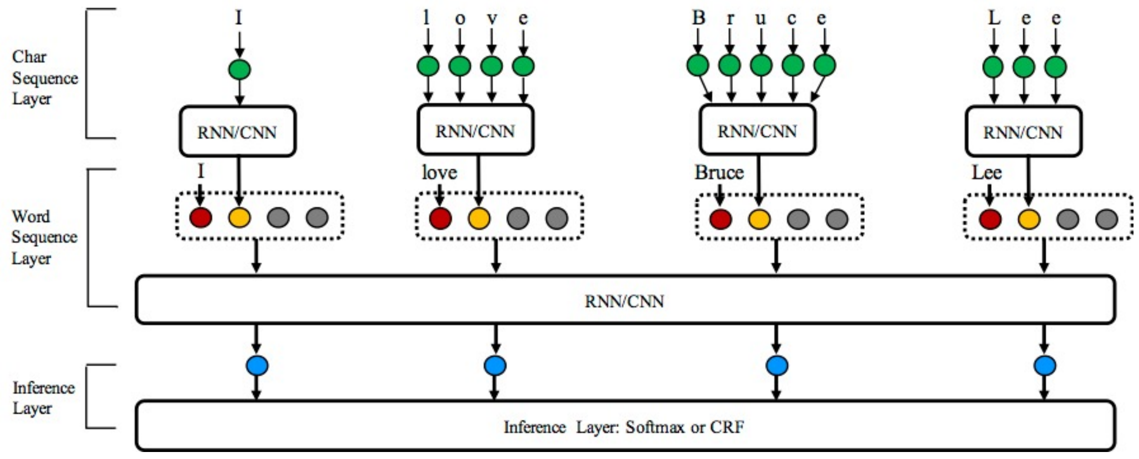


Figure 3.1: Model Architecture

3.4.2 Embedding Layers

- **Word2vec**: Word2vec is a method to efficiently create word embeddings and has been around since 2013.It allow user to compute continuous vector representations of words from very large datasets in less time but also improve the semantic regularities of learned words by taking advantage of word off-set technique.

- **FastText**: FastText is an open-sourcing library which combines concepts from other successful models such as Bag-Of-Word, N-gram with the help of new extension called subword information to tackle problem. FastText is an extension of the word2vec model.It represents each word as an n-gram of characters. This helps capture the meaning of shorter words and allows the embeddings to understand suffixes and prefixes. FastText can be said to outperform other deep learning based methods in terms of training times by significantly decreasing it from several days to just a few seconds while increasing the accuracy on many standard problems, such as sentiment analysis.

- **BERT**: A transformer architecture is an encoder-decoder network that uses self-attention on the encoder side and attention on the decoder side. BERT BASE has 12 layers in the Encoder stack while BERT LARGE has 24 layers in the Encoder stack.This model takes CLS token as input first, then it is followed by a sequence of words as input. It then passes the input to the above layers. Each layer applies self-attention, passes the result through a feedforward network after then it hands off to the next encoder

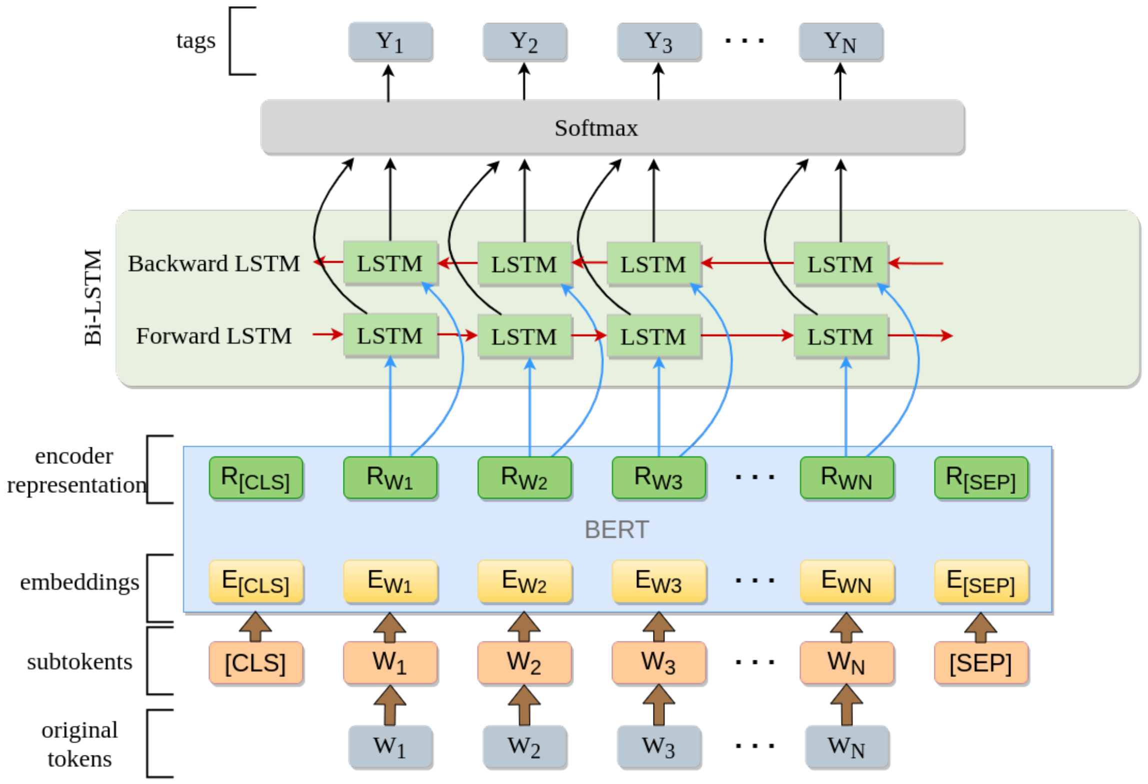


Figure 3.2: BERT+BiLSTM Architecture of our proposed system

3.4.3 Word Sequence Layer

- **Bi-LSTM**: Bi-LSTM means bidirectional LSTM, which means the signal propagates backward as well as forward in time. You can also apply this architecture to other RNNs. A Bi-LSTM calculates the input sequence from the opposite direction to a forward hidden sequence and a backward hidden sequence . The encoded vector is formed by the concatenation of the final forward and backward hidden unit outputs. where is the output sequence of the first hidden layer.

  In this project we used a basic LSTM model. A single Bi-LSTM with 300 hidden units is used. The word embeddings of 300 dimensions are passed through this layer. The output at each time step is subjected to two dense layers of size 100 and 1. A dropout of 0.5 is used in the recurrent layer.

- **CNN**: The CNN component is used to induce the character-level features. For each word, the model employs a convolution and a max-pooling layer to extract a new feature vector from per-character feature vectors such as character embeddings and (optionally) character type. The output is then subjected to two dense layers. The time axis corresponds to the number of characters in the word.

- **MultiCNN**: This combines four layers of CNN layer. The word embeddings generated by this multi-CNN network is then concatenated with 300-dimension learnable word embeddings as used in previous models. Finally, the concatenated representations are passed through a dense layers.

3.4.4 Baselines and Evaluation metrics

We include baseline models from Joshi et al. (2020); Jamatia et al. (2019). Joshi et al. (2020) used a character, word, and sub-word representation with

CNN, CNN-LSTM, and simple LSTM layer. The sub-word representation with the LSTM layer performed the best. Jamatia et al. (2019) used pre-trained word embedding (GloVe) with LSTM layer (Figure 4a) and character level Recurrent Neural Network (RNN) with Conditional Random Fields (CRF) classifier (Figure 4c). The models are trained on the training dataset and evaluated by predicting the labels for the test dataset.

Evaluation Metrics: We evaluate and compare every model's performance in terms of precision, recall, F1-score, and accuracy for highly imbalanced label distribution, where their definitions considered are as given below. Accuracy and weighted average F1-score are used as primary evaluation metrics.

3.5 Experiments and results

Character level RNN and CRF-based model (Figure 4c) obtain an accuracy of 85.33% and 86.35% on the language pairs Bengali-English and Hindi-English, respectively. BERT and mBERT are used as pre-trained embedding models. BERT models are trained for maximum 150 epochs depending on the learning rate. The architectural diagram of BERT model is shown in Figure 4b. BERT based experiments are performed on Google's Colabi. PyTorch deep learning library is used to implement the models. We also use HuggingFace's transformers to fine-tune pre-trained BERT based models.

We measure the performance of each model using accuracy, precision, recall and F1 scores. Figure 5, 6 and 7 show the overall system performance of class level on the ICON POS (BN-EN) dataset.

For ICON SAIL (HI-EN) dataset, we obtain 92.48 accuracy on BERT base cased embedding with Bi-LSTM layer (Table 7). The results are obtained after training for 25 epochs on the data sets. Increasing the number of epochs to 30 and 50 provided no improvements. FastText model with concatenation of Multi CNN and Bi-LSTM

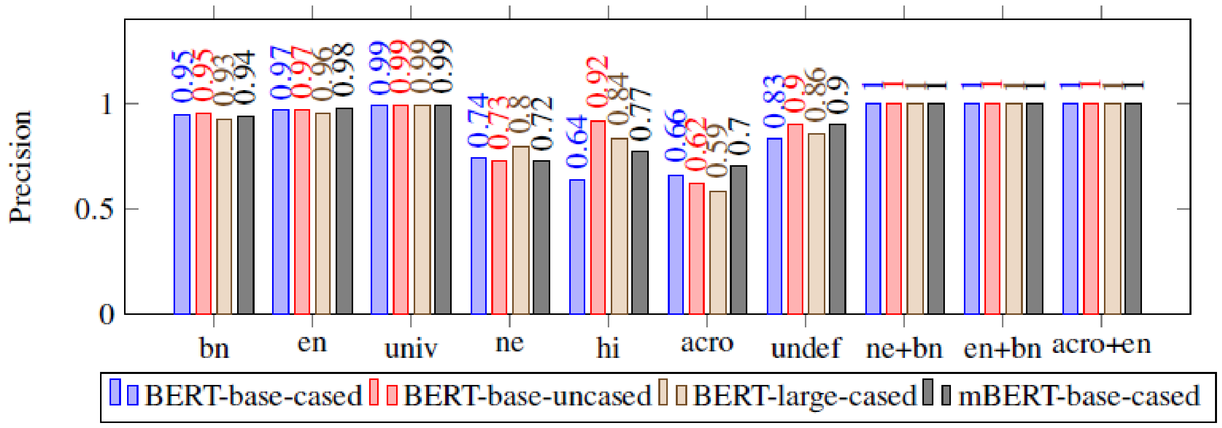


Figure 3.3: Graphical comparison of Precision for all experiments on ICON POS (BN-EN) dataset

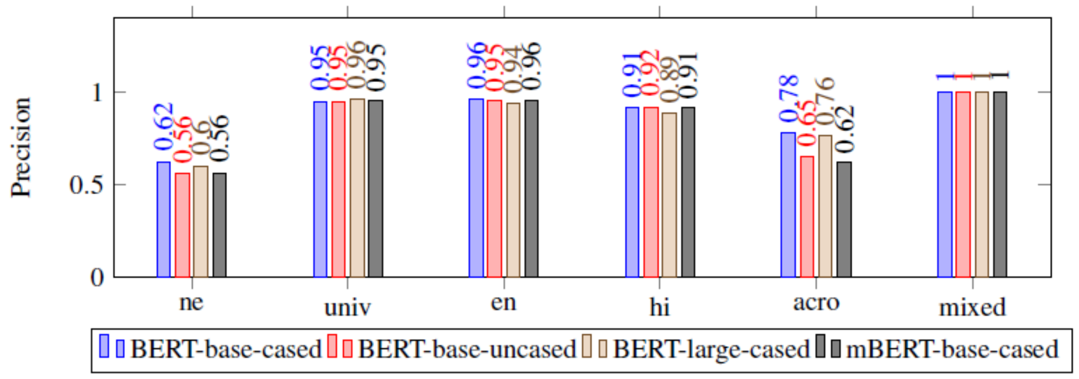


Figure 3.4: Graphical comparison of Precision for all experiments on ICON POS (HI-EN) dataset

layer (Figure 4d) has an accuracy of 94.09 and emerges as our best performing model for ICON SAIL (BN-EN) dataset. The result for the FastText model is shown in Table 8. The model has 4 CNN layers with 20 characters hidden dimension and 200 hidden dimension layer. Also, dropout is 0.5, and the learning rate is 0.015.

Despite the far smaller pre-trainings, the contextual embedding on the code-mixed corpus has better results. BERT model outperforms on the ICON POS (BN-EN), ICON POS (HI-EN) and ICON SAIL (HI-EN) datasets but could not perform well on the ICON SAIL (BN-EN) dataset.

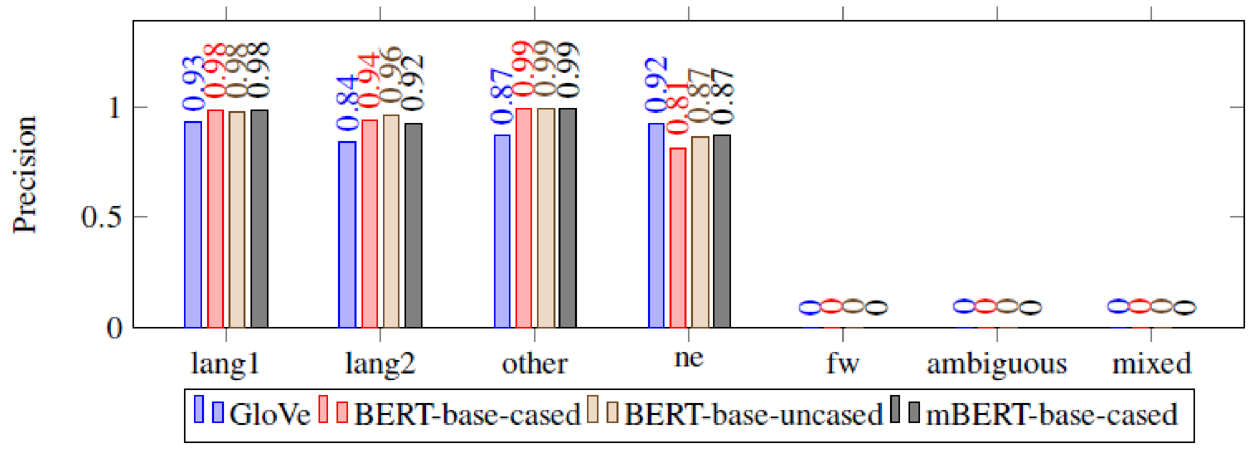


Figure 3.5: Graphical comparison of Precision score for all experiments on LinCE (HI-EN) dataset

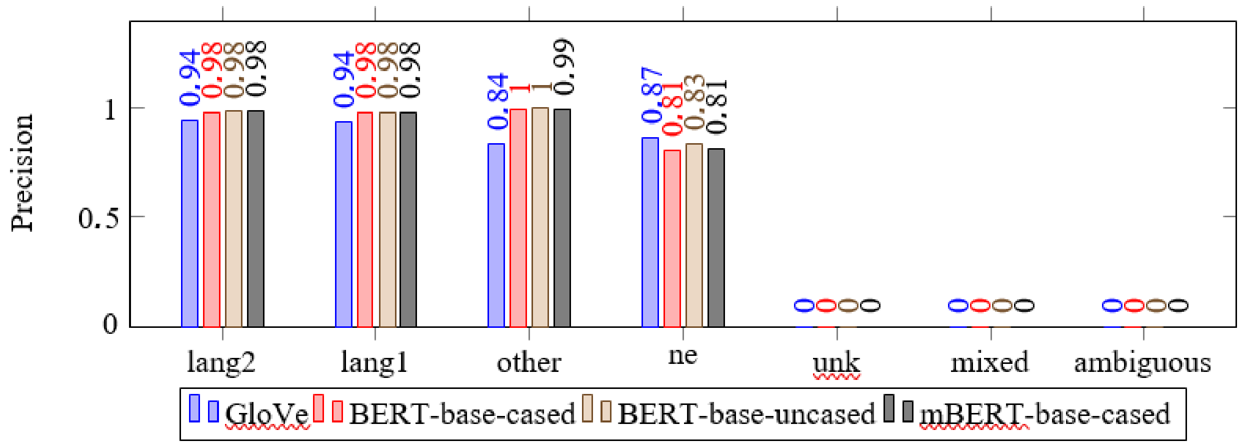


Figure 3.6: Graphical comparison of Precision score for all experiments on LinCE (ES-EN) dataset

The improvements in accuracy and F1 score can be attributed to reduction in the number of token mismatches using the BERT word embeddings. The confusion matrices (Figure 17) for both the models show that many tokens which are wrongly classified in GloVe model (Figure 17a), are not present in the BERT model (Figure 17b). Our model does not predict any intra- word code-mixed data like Facebooke (ne+bn suffix tag).

This is one type of code-mixed data where mixing occurs within a word itself. BERT uses WordPiece technique to diminish unknown token occurrences significantly which must have played a role in the success and dealing with the out-of-vocabulary issue. The inability of the baseline system to capture context information was one of its major flaws. Some words with identical spellings in different languages retain the same tag, as seen in the results. For example,

- **Original**: Ota cmplt hole 1 ti montro jop korte hobe 25 bar.
- **Desired translation**: When it is complete, you have to chant 1 mantra 25 times.

Here, the baseline model predicts cmplt as a bn tag, hole, and bar as an en tag, but hole and bar are Bengali words as well with different meaning than in English. The baseline models could not understand the context and failed. Our model distinguishes spelling variations of complete keywords and predict en tags, as well as analyze the context of two words and predicts the correct tags.
Another aspect may be the inconsistency of the labels. Some of the words like upokrito (bene- fitted), golata (The throat), and janle (If you know) are wrongly labelled as en in the gold standard dataset. Our model correctly marked them as bn. As native speakers, we are confident that our predicted tags are correct, as these all are Bengali words and have been used in a Bengali con- text. On the other hand, it is also noticed that sometimes our model predicts a word as en, but the gold standard dataset marked as bn like gender, are, on, change, to, be, expectation. These all are English words and have been used in an English context. We can see that the LinCE dataset is correctly annotated in these three datasets, which helped our model obtain a high score above the other two datasets.

3.5.1 Comparison with Baseline

We compare the performance of our proposed model with that of baseline Joshi et al. (2020) and Jamatia et al. (2019) models in Figure 3.6, 3.7 and 3.8, 3.9 respectively. We observe that our proposed model performs better for both the datasets. The overall performance margin for ICON POS (BN- EN) data is 7.14% and for ICON POS (HI-EN) data is 6.66%. On the other hand, for ICON SAIL (HI-EN) dataset, we have received comparatively less performance gain (0.53)%. For LinCE (HI- EN) and LinCE (ES-EN) dataset our model outperform the baseline model.

Embedding	Model	Accuracy$_{ALL}$(%)	Accuracy$_{CM}$(%)
Character Level RNN	CRF	85.33	83.53
GloVe	Bi-LSTM	88.60	83.92
BERT Base Cased	Bi-LSTM	95.18*	94.68
BERT Base Uncased	**Bi-LSTM**	**95.74***	94.72
mBERT Base Cased	Bi-LSTM	95.42*	94.92
BERT Large Cased	Bi-LSTM	94.42*	92.77

Table 3.7: Comparison of baseline with best proposed model on ICON_POS (BN-EN) Data

Embedding	Model	Accuracy$_{ALL}$(%)	Accuracy$_{CM}$(%)
Character Level RNN	CRF	86.35	79.73
GloVe	Bi-LSTM	87.84	85.33
BERT Base Cased	**Bi-LSTM**	**93.50***	91.96
BERT Base Uncased	Bi-LSTM	92.81*	91.07
mBERT Base Cased	Bi-LSTM	92.84*	91.51
BERT Large Cased	Bi-LSTM	91.86	91.49

Table 3.8: Comparison of baseline with best proposed model on ICON_POS (HI-EN) Data

3.6 Conclusion

In this paper, we tackle the most fundamental problem of any code-mixed downstream task: the problem of language identification. The difficulty of identifying languages in code-mixed data is determined by the data type and code-mixing behavior. We study

Embedding	Model	Accuracy$_{ALL}$(%)	Accuracy$_{CM}$(%)
Word2vec	Bi-LSTM	92.41	92.21
Word2vec	CNN	91.91	91.86
Word2vec	Multi CNN	92.07	91.94
FastText	BiLSTM	92.26	92.17
FastText	CNN	92.05	91.27
FastText	Multi CNN	92.17	92.02
BERT Base Uncased	**Bi-LSTM**	**93.04**	92.35
mBERT Base Cased	Bi-LSTM	92.89	92.67

Table 3.9: Comparison of baseline with best proposed model on ICON_SAIL (HI-EN) Dataset

Embedding	Model	Accuracy$_{ALL}$(%)	Accuracy$_{CM}$(%)
Word2vec	Bi-LSTM	79.28	77.66
Word2vec	CNN	79.52	79.34
Word2vec	Multi CNN	78.53	76.16
FastText	Bi-LSTM	80.47	84.04
FastText	CNN	75.37	78.26
FastText	Multi CNN	76.88	77.51
BERT Base Uncased	**Bi-LSTM**	**83.09**	82.17
mBERT Base Cased	Bi-LSTM	83.16	82.29

Table 3.10: Comparison of baseline with best proposed model on ICON_SAIL (BN-EN) Dataset

Embedding	Model	Accuracy$_{ALL}$(%)	Accuracy$_{CM}$(%)
CharRNN	CRF	85.91	80.56
GloVe	Bi-LSTM	90.03	87.4
BERT Base Cased	Bi-LSTM	95.8*	94.69
BERT Base Uncased	**Bi-LSTM**	**96.1***	94.79
mBERT Base Cased	Bi-LSTM	96.06*	93.77

Table 3.11: Comparison of baseline with best proposed model on LinCE (HI-EN) development data

Embedding	Model	Accuracy$_{ALL}$(%)	Accuracy$_{CM}$(%)
CharRNN	CRF	92.24	91.89
GloVe	Bi-LSTM	91.42	90.3
BERT Base Cased	Bi-LSTM	98.16*	97.7
BERT Base Uncased	**Bi-LSTM**	**98.22***	97.96
mBERT Base Cased	Bi-LSTM	98.08*	97.93

Table 3.12: Comparison of baseline with best proposed model on LinCE (ES-EN) development data

the problem with extensive experiments using multiple models and architectures with different representations of the input texts. Our Bi-LSTM model on top of BERT neural representations of code-mixed data emerged as the best performing model. For the language pairings analysed, the deep learning model with fine- tuned pre-trained word embedding model outperforms the classic CRF approach. Just changing the input representation helps achieve the best results. In the future, it would be fascinating to investigate and compare the code-mixing behavior of data from various sources across other low- resource (Indian) language pairs.

Chapter 4

Code Mixed Information Retrieval

4.1 Introduction

In today's world, social media is nearly everywhere. Such expansion necessitates for automatic information processing, which presents a number of issues. The majority of social media content is informal. Furthermore, when discussing Indian social media, individuals frequently prefer to employ Roman transliterations of their local languages with English embedding. As a result, information retrieval (IR) on such Indian social media data is a difficult problem when the documents and queries are a mixture of two or more languages written in either native scripts or Roman transliterated form.

People have become an active component of a digital ecosystem connected with a wide range of gadgets as a result of Web 2.0. Unlike in the past, when the bulk of users would passively browse for information from helpful scholarly sites, today's users actively create, distribute, and tag multidimensional content on the Web. They do so formally at times, but more often informally on various social networking sites like Facebook, Twitter, Instagram, and so on. The textual content in various social media differs in size, format, and character from traditional formal text content. It comprises a lot of personal rambling, slang, the usage of numerals (even within textual words in SMS texts), a combination of multiple scripts (English and non-English), different languages even utilising a single script, and so on. Retrieving meaningful information from data in this format thus becomes an important research area.[8, 9]

4.2 Related Work

When submitting inquiries or publishing blogs in text, users seldom follow any traditional linguistic standards to build sentences(s). Such fully/partially complete sentence(s) are typically created in colloquial languages that are regularly transliterated. The script's orthography is dependent on word pronunciation, and content written in a native language but using a non-native script does not adhere to any conventional spelling norms. The practise of phonetically transforming a language's words into a foreign or non-native alphabet is known as transliteration. Transliteration, particularly into Roman script, is becoming increasingly popular on the Internet, not just for papers but also for a number of other applications.[10]

Till now, quite a substantial amount of works have been done on transliterated search and fields like Hindi song lyrics retrieval, where the whole sentences are in Indian languages but written in Roman script (transliteration). Also with the advent of Web 2.0, the number of Web users and the volume of the Web content have grown at an exponential rate. The majority of these users are not simply consumers of information, but also producers of it. People here communicate in informal languages. On social media platforms, people not only communicate using transliterated texts but mostly Code-Mixed texts. Since not much work has been done on Code Mixed Information Retrieval (CMIR), we focus on this task in our research project.

Under the CMIR configuration, the IR task is to search for documents written in several languages and rank them based on the query provided. CMIR on social media texts adds new hurdles to the IR endeavour. First, determining the language of each query term for a particular query is a challenging task. Code-Mixed search across Code-Mixed tweets with a short document length. The presence of spelling variants in transliterated query phrases is the second problem.

4.3 Dataset Creation

Code-Mixed dataset is not readily available on the internet for accomplishing our task. Thus a substantial amount of time has been spent on collecting data to build our initial dataset. No place can be better for code-mixed data than social media platforms.

We crawled through Facebook public pages and public groups to collect the required data. The content of original posts was treated as queries and the comments were treated as the corpus from which relevant data needs to be retrieved. E.g.: In a public Bengali community group for people living in Mumbai, one might ask for a good Bengali cuisine nearby as a post, and the task is to find out relevant answers from all the comments. Similarly, the same process was done for public Telegram groups.

Initially, we made a small dataset consisting of 19 queries and 1942 documents on which our preliminary experimentations were done. Then we expanded this to build our main dataset consisting of 50 queries and 145278 documents. All the experimentations and results shown in this paper are based on this dataset. The data statistics are shown below.

4.3.1 Code-mixing Metrics

Total sentences: 145278

Total code-mixed sentences: 84694

	Data	Query
all sentences	14.717	27.424
code-mixed sentences	25.244	29.175

Table 4.1: Code-Mixing Index (CMI)

	Data	Query
all sentences	0.523	0.951
code-mixed sentences	0.898	1.012

Table 4.2: Multilingual Index (MI)

	Data	Query
all sentences	0.198	0.277
code-mixed sentences	0.340	0.294

Table 4.3: Integration Index (II)

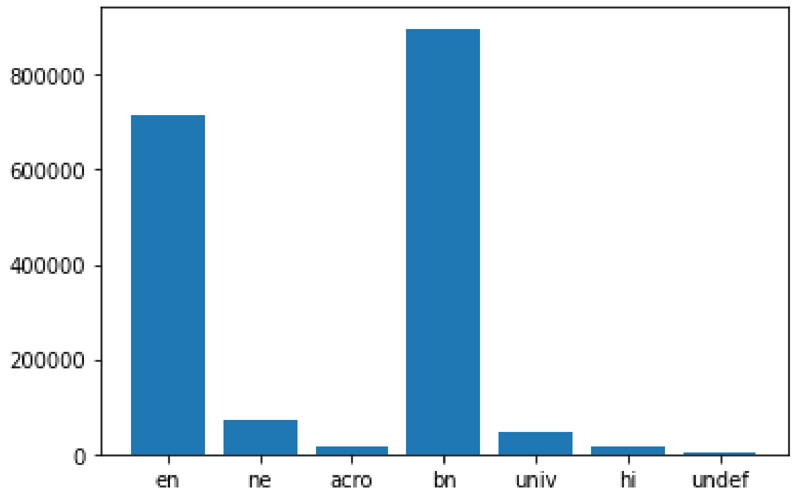
4.3.2 Language Tag Distribution

We run the Language Identification model on our data corpus to find the language token of each term. Following are the language tags:

- **en**: English
- **bn**: Bengali
- **hi**: Hindi
- **ne**: Name Entity

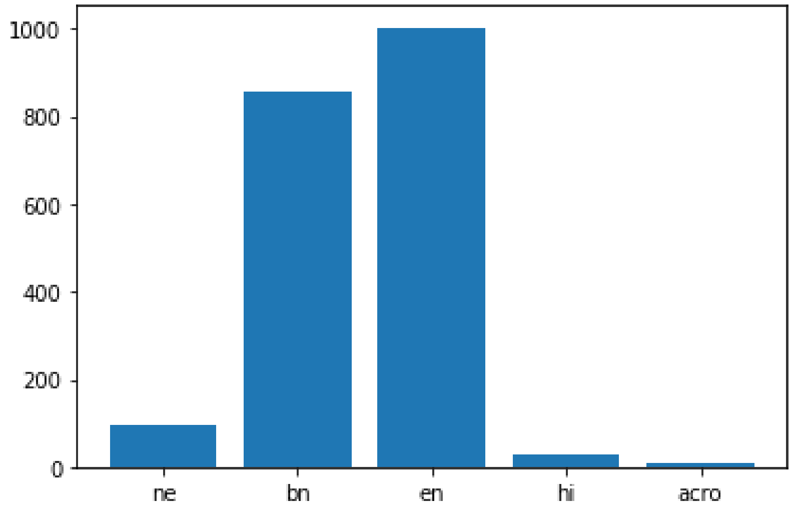
- **acro**: Acronym
- **univ**: Universal
- **undef**: Undefined

We used our own Language Identification module. To check the accuracy we took random few sentences and evaluated those and got **97.7% accuracy**.



.4

Figure 4.1: Tag distribution in Data



.4

Figure 4.2: Query

Figure 4.3: Tag distribution in Query

4.3.3 Sentence Length Distribution

Below is the plot of the distribution of number of words in a sentence versus the number of documents.

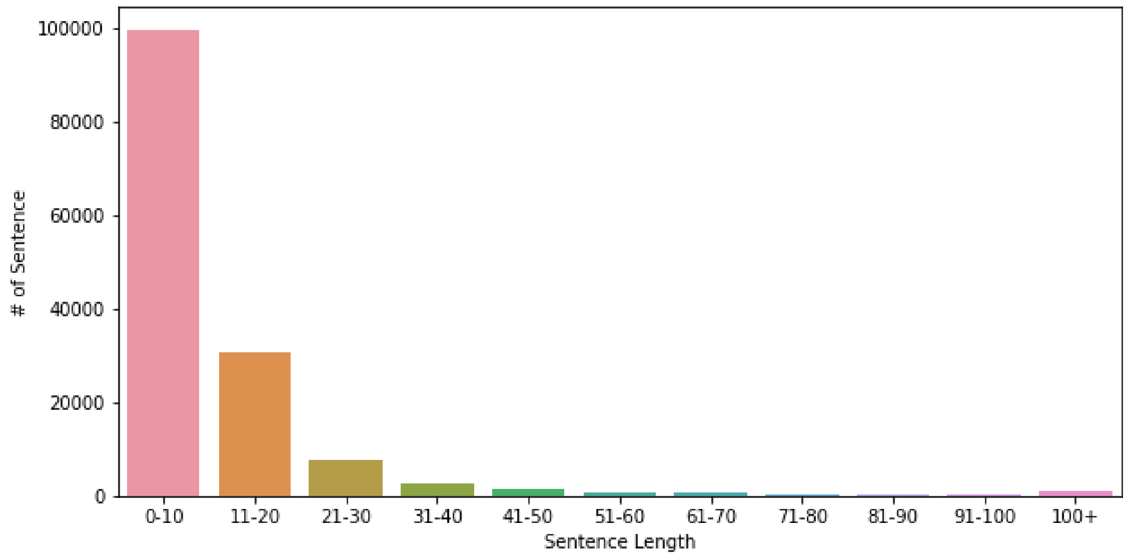


Figure 4.4: Distribution for Data

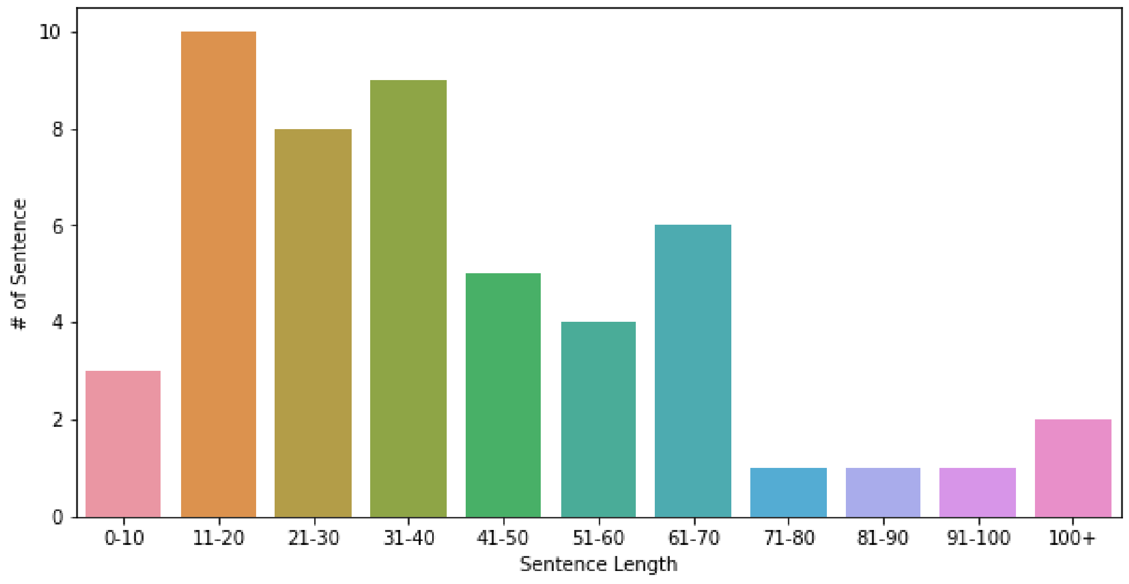


Figure 4.5: Distribution for Query

4.4 Methodology

4.4.1 Phonetic Matching Algorithms

Despite being phonetically identical, user-generated content strings may differ at the character or token level, especially in transliterated language. For example, the words “dhanyabad” and “dhanyvad” are phonetically identical, yet their string representations differ. To match such strings, phonetic algorithms deal with this issue.[11] There are several phonetic encoding algorithms, such as Soundex, Phonix, and Editex. Soundex and Phonix employ one-to-one code mapping, whereas the other algorithms use one-to-one, many-to-one, and/or many-to-many as needed. [1]

Soundex [1]

It’s the most widely used phonetic coding technique. It was created to extract phonetically similar English surnames. It employs a four-character code for encoding, with the first letter followed by three digits. The scheme’s design is mostly focused on the sound of consonants. The English language is divided into seven divisions based on sound alphabets. The phonetic coding of letters in these groups is encoded with six numbers (1-6), while all other English alphabets are preserved in a seventh group (with a ’0’ tag), which is eliminated in the final encoding.

Example	
Bengali Word	Soundex Code
Phool	P400
Phul	P400
Ful	F400
Full	F400
Fool	F400

Table 4.4: Example of Bengali word and Soundex code

Phonix [1]

It is another phonetic algorithm designed for the retrieval of similar-sounding words (mostly names) with different spellings. Except for the replacements, this technique is similar to Soundex in theory. The set of encoding codes differs somewhat.

Example	
Bengali Word	Soundex Code
Phool	F400
Phul	F400
Ful	F400
Full	F400
Fool	F400

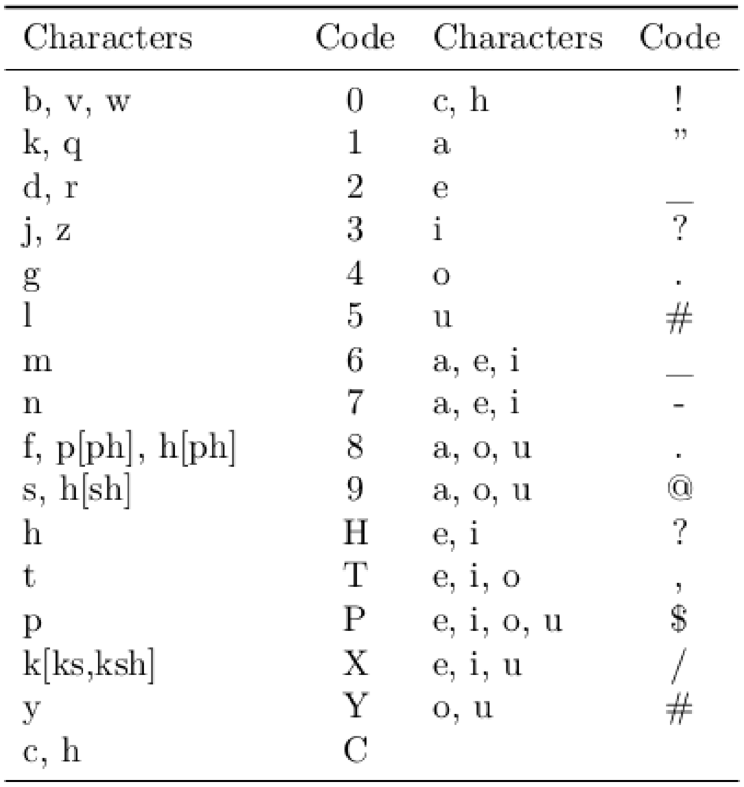
Table 4.5: Example of Bengali word and Phonix code

Hindex [1]

To assess dissimilarity between two words, Hindex employs the core notion of character wise mapping from Soundex and Editex, as well as Levenstein edit distance as an approximation string matching technique. It is a personalised set of principles for phonetic encoding of transliterated Hindi/Bengali terms based on observations for transliterated Hindi/Bengali terms in Roman. These principles are drawn from the fundamental notions of Soundex, Phonix, Editex, and phonetic sounds: vowels and consonants. Except for a handful, the set of phonemes for a certain phrase is formed by combining graphemes from a consonant alphabet(s) with a vowel(s).

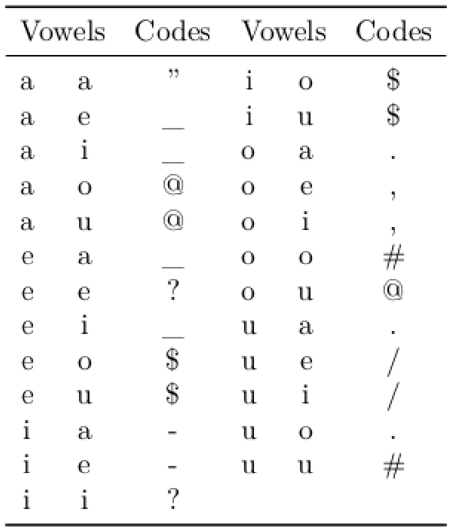
Steps:

- All the graphemes are encoded with the corresponding codes defined for its phone group in the Table 4.6. Often two or more similar vowels are used to emphasize the presence of vowel-modifiers in Hindi. Table 4.7 represents coding schemes for all possible vowel-pairs used for this purpose.



Characters	Code	Characters	Code
b, v, w	0	c, h	!
k, q	1	a	”
d, r	2	e	_
j, z	3	i	?
g	4	o	.
l	5	u	#
m	6	a, e, i	_
n	7	a, e, i	-
f, p[ph], h[ph]	8	a, o, u	.
s, h[sh]	9	a, o, u	@
h	H	e, i	?
t	T	e, i, o	,
p	P	e, i, o, u	$
k[ks,ksh]	X	e, i, u	/
y	Y	o, u	#
c, h	C		

Figure 4.6: Hindex code



Vowels		Codes	Vowels		Codes
a	a	”	i	o	$
a	e	_	i	u	$
a	i	_	o	a	.
a	o	@	o	e	,
a	u	@	o	i	,
e	a	_	o	o	#
e	e	?	o	u	@
e	i	_	u	a	.
e	o	$	u	e	/
e	u	$	u	i	/
i	a	-	u	o	.
i	e	-	u	u	#
i	i	?			

Figure 4.7: Code for vowels pair

- Codes are assigned from the first character unlike Soundex where codes are assigned from the second character onwards.

- Sometimes a sequence of letters represent a single phone leading to a single code for the the sequence such as ‘ph’ or ‘ksh’. The sequence is identified with 2-characters lookahead (more than 2 lookahead is very unlikely). All the individual letters in the same sequence get the same code according to Table 4.6 in the first pass.
- Code assignment for the letters need not follow mutually exclusive group property as few characters sometimes sound independently, at other times act as sound-modifiers.
- All the identical codes placed besides are replaced with single one, hence the codes generated in previous step are normalized.

Example	
Bengali Word	Hindex Code
Phool	8#5
Phul	8#5
Ful	8#5
Full	8#5
Fool	8#5

Table 4.6: Example of Bengali word and Hindex code

4.4.2 Zero Shot learning

Because of the shortage of ample relevance judgements for languages other than english, an approach of zero shot cross-lingual ranking is used. The basic idea here is to use the relevance judgement data of a language with ample data (English for instance) to train and validate a neural ranking model and then use this model to test for data in different languages. Here the term zero shot [12] signifies that the model at testing has not seen the data corresponding to test language during its training phase.

More formally, we can state this as follows: The typical information retrieval process contains a set of tuples S containing **<q, d, r>** tuples representing (query, document, relevance). We split the entire relevance judgement set into two parts S_{train} and S_{test}. Both the sets here are strictly disjoint. The S_{train} is then used to tune a ranking function $R_{\text{S}_{\text{train}}}(q, d) \in R$, which is then tested on S_{test}. In zero shot learning, we have two relevance feedback sets S and T, where S is the relevance feedback corresponding to the data for the language having greater quantity of data (English in our case) and T is the relevance feedback corresponding to our target data (English-Bengali code mixed data in our case). Here instead of splitting the set S into training and testing, we utilize the entire set S for training the ranking function and use T containing data in the second language for testing.

We have used the following neural ranking models for reranking the results of BM25 model:

1. **KNRM** [13] is a neural model for document ranking that is built on kernels. K-NRM employs a translation matrix to model word-level similarities via word embeddings, a kernel-pooling approach to extract multi-level soft-match features, and a learning-to-rank layer to merge those features into the final ranking score, given a query and a set of documents.
   Let $\{q_1, q_2, ..., q_{\text{n}}\}$ denote the set of queries and $\{d_1, d_2, ..., d_{\text{m}}\}$ denote the set of documents. The aim is to learn a ranking function $f(q, d) \in R$, which gives the relevance score for each query document pair.
   Following steps highlight the procedure adopted by the KNRM model.

   - Each word is converted into a L dimensional vector ($L = 300$) using a suitable word embedding technique (Fasttext used in our case). Here, t

denotes the term and v_{t} denotes its embedding vector of dimension 300.

$$v_t = embed(t) \tag{4.1}$$

- A translation matrix is generated which is a 2D cosine similarity matrix M, where M_{ij} denotes the similarity between the i^{th} word of the query and j^{th} word of the document. Here, $v_{\mathrm{t_iq}}$ denotes the embedding vector corresponding to the i^{th} term of the query and $v_{\mathrm{t_jd}}$ denotes the embedding vector corresponding to the j^{th} term of the document.

$$M_{ij} = cos(\vec{v_{t_i}}q, \vec{v_{t_j}}d) \tag{4.2}$$

- The model then uses kernels to convert the word-word interactions of the translation matrix into query-document ranking features vector of dimension k corresponding to each row of the translation matrix. The below formula represents the k^{th} kernel corresponding to M_{i} of the translation matrix.

$$K_k(M_i) = \sum_j \exp\left(\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}\right) \tag{4.3}$$

- Next step is the kernel pooling wherein a vector of dimension k is created corresponding to each row of the translation matrix. For each row M_{i} of the translation matrix $K(M_{\mathrm{i}})$ is a vector consisting of all the kernels from 1 to k as its elements corresponding to its k dimensions.

$$\vec{K}(M_i) = \{K_1(M_i), \ldots, K_K(M_i)\} \tag{4.4}$$

- The query-document ranking features are then generated using the matrix generated after kernel pooling. Here Φ represents the k-dimensional vector representing the query-document ranking features.

$$\phi(M) = \sum_{i=1}^{n} \log \vec{K}(M_i) \tag{4.5}$$

- The ranking features are finally combined using a ranking layer to generate the final ranking score. Here, $f(q, d)$ represents the final ranking function which takes query (q) and document (d) as its inputs and uses w as weights and b as the bias.

$$f(q, d) = \tanh\left(w^T \phi(M) + b\right) \tag{4.6}$$

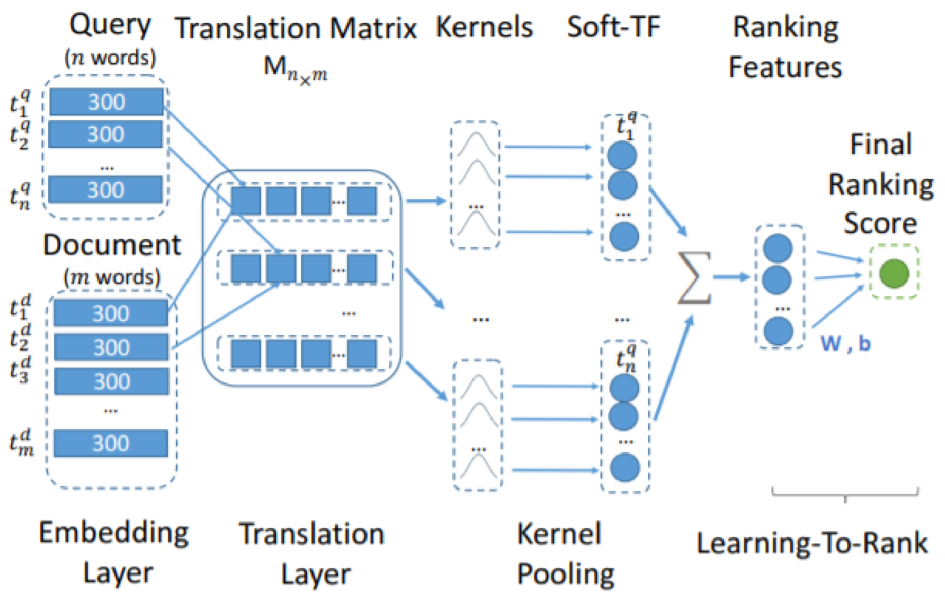


Figure 4.8: KNRM

2. **Vanilla Bert** [7]: This rather simple model comprises a Bert Base Uncased model with input in two parts. The segment A of the bert model consists of the query and segment B consists of the document. The segments are preceded by a $[CLS]$ tag and the segments are separated by $[SEP]$ tags. The output vector corresponding to the $[CLS]$ tag is followed by a linear combination layer which

finally calculates the ranking score. The complete neural ranking model is fine-tuned on the complete retrieval ranking task using the pairwise cross entropy loss and Adam optimizer. The following figure demonstrates the vanilla bert model.

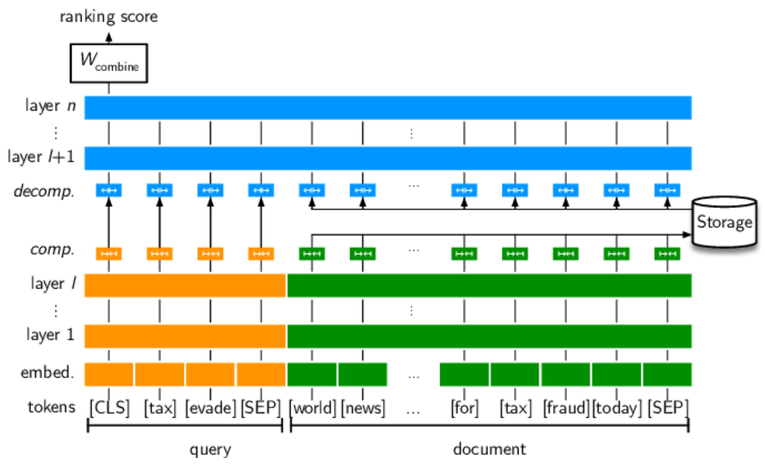


Figure 4.9: Vanilla Bert

3. **MonoT5**: MonoT5 [14] is a pointwise reranking model and we have used this model to rerank the documents generated by BM25. MonoT5 uses T5, which is a pre-trained sequence-to-sequence transformer model. In this model, all target tasks are cast as sequence-to-sequence tasks, with query and document pair taken as input and the relevance as a binary output. The task is to fine-tune the model to predict “true” or “false” depending on whether the document is relevant for the query or not. Here “true” and “false” are our target and the task can be seen as a binary classification task.

   We use a softmax solely on the logits of the ”true” and ”false” tokens at inference time to compute probabilities for each query–document pair (in a reranking setting). The documents are reranked based on the probabilities assigned to the ”true” token. Note that, although H0 employs a corpus that has been supplemented by document expansion, R0 documents are original texts that do not contain the expected questions.

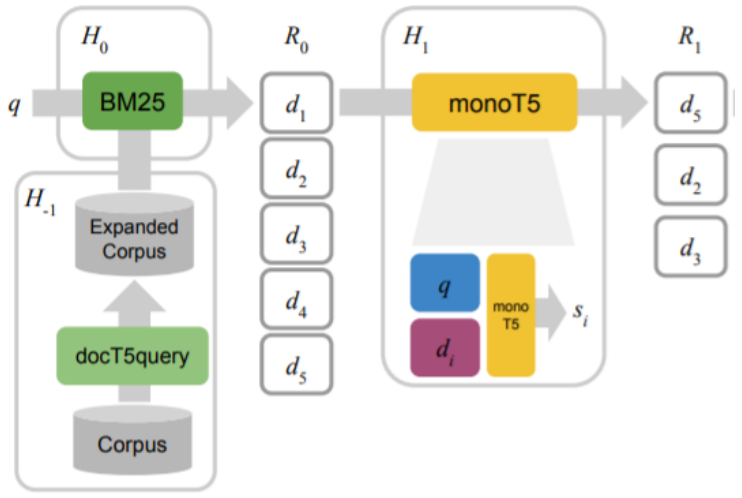


Figure 4.10: MonoT5

4. **DeepCT**: DeepCT [15] stands for Deep Contextualized Term weighting framework. DeepCT is made up of two essential components: generating contextualized word embeddings through BERT, and predicting term weights through linear regression.

   - **Contextualized word embedding generation**: Bert is used for this phase which is a contextualized language model. BERT employs an attention mechanism in which a word gradually absorbs context information depending on the attention it pays to every other word in the same text.
   - Term weight prediction: The contextualized word embeddings generated in the previous step are used to calculate the word importance score by performing the per-token regression task using the mean square error (MSE) loss. DeepCT combines word features into term weights linearly.

$$\hat{y}_{t,c} = \vec{w}T_{t,c} + b \tag{4.7}$$

Here, $T_{t,\ c}$ denotes the embedding of the token t in text c. The generated term weight lies in the range $[0, 1]$.

For unseen words, BERT generates sub-words corresponding to the word and the term weight of the first subword is taken as the weight of the complete word.

In DeepCT, the entire model is trained from end-to-end, from BERT to regression layer, thus, fine tuning BERT and learning the regression layer from scratch. In a language environment, DeepCT is a generic framework for learning word weights. It is possible for DeepCT to learn different meanings of word significance depending on how the ground truth term weights are specified. Projected term weights are also useful for a variety of purposes depending on the job.

DeepCT-Index uses DeepCT to weight passage phrases before storing the weights in a traditional inverted index. In our work we have used DeepCT to index terms and then use BM25 to retrieve documents for queries.

4.4.3 Information Retrieval Models

Information Retrieval is the process of searching relevant documents based on query terms that should satisfy a user's information need. A retrieval system has different components such as tokenization, indexing, similarity measuring for the relevance etc. In the tokenization, from all the documents in the corpus sentences are split into words called tokens and simultaneously punctuations are removed. Based on indexing schemes (one word or bi-word) tokens are indexed. The Index maintains tokens entries along with other details as posting list. While measuring the weight, similarity is measured against given query and documents in the corpus. Several weighing models have been developed like boolean model, tf-idf model, vector space model and language model. The working principle of boolean retrieval model is based on functionality of logical operators (AND, OR and NOT) on query and document terms.

The tf-idf model is bag-of-words model where retrieval depends on the frequency of query terms into the query itself and in the documents. In the vector space model, query and documents are mapped into vector form. Then cosine similarity between query and document vectors are computed and based on similarity score documents are retrieved. The language model deals with context information where retrieval is based on consecutive terms in the query as well documents (such as word 2-gram, 3-gram etc.). [1]

In our case, we used some of the most popular IR models. Some of them are listed below:

- **BM25**: It is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document. It is a family of scoring functions with slightly different components and parameters.
- **TF_IDF**: It is a model with the tf*idf weighting function, where tf is given by Robertson's tf and idf is given by the standard Sparck Jones' idf.
- **BB2**: It is a Divergence-from-randomness (DFR) model with Bose-Einstein model for randomness, the ratio of two Bernoulli's processes for first normalisation, and Normalisation 2 for term frequency normalisation.
- **InL2**: It is a Divergence-from-randomness (DFR) model with inverse document frequency model for randomness, Laplace succession for first normalisation, and Normalisation 2 for term frequency normalisation.
- **DPH**: It is a new hyper-geometric Divergence-from-randomness (DFR) model that employs Popper's normalisation (parameter free).
- **LM**: It is Hiemstra's language model.

4.5 Experimental Setup

4.5.1 Terrier / PyTerrier

Terrier

Terrier [16] is a highly adaptable, efficient, and effective open-source search engine that can be easily deployed on enormous collections of information. Terrier has cutting-edge indexing and retrieval capabilities, making it a perfect platform for the quick creation and assessment of large-scale retrieval applications. It is an open-source framework for text retrieval research and experimentation that is thorough, adaptable, and transparent. Standard TREC and CLEF test collections lend themselves well to research. Terrier is built in Java and was created by the Information Retrieval Group at the University of Glasgow's School of Computing Science. We did most of our indexing, retrieval and evaluation tasks using the Terrier IR platform. This made our tasks more efficient and streamlined.

Pyterrier

Pyterrier [17] is a platform that exposes the terrier functionality to the declarative form of python making it easier and convenient to conduct information retrieval experiments. It uses the Java-based Terrier information retrieval platform internally to support indexing and retrieval operations. Pyterrier deals with pandas dataframes which makes creation and manipulation of data simple and main emphasis is on experimentation. It also enables creations of complex pipelines and provides out of the box neural models to use in pipelines or separately. The evaluation metrics are calculated by the pytrec_eval library, a Python wrapper around the widely-used trec_eval evaluation tool. [1]

[1] http://terrier.org/

4.5.2 Trec Format

TREC is an acronym for the National Institute of Standards and Technology's Text REtrieval Conference. Each document must include begin document and end document tags in order for the indexer to know where the document boundaries are inside files. These tags, which are comparable to HTML or XML tags, constitute the format for TREC documents.

A TREC file is roughly written in this following format:

```
<DOC>
<DOCNO> document_number </DOCNO>
<TEXT>
Index this document text.
</TEXT>
</DOC>
```

So the data and the queries were all converted into the required format for Terrier using a python script, which were thereby used for all the subsequent downstream tasks. Data is stored in separate files where each file contains a single document, whereas queries are all stored in a single TREC file.

Example

Sample data file:

```
<DOC>
<DOCNO>1</DOCNO>
<HEAD> </HEAD>
<BODY>
apparently there is no partial lockdown at germany ny nj london as they are highly
```

worried about their economic slowdown life is precious there but not more than trade i am updated on this lockdown issue not by media or newspapers or internet my friends family members who are right now based at those above mentioned countries localities informed about their condition whereas in india despite of huge threat of recession we have been maintaining absolute lockdown as our country wants to save lives at any cost india s fight to deal with this corona catastrophe is really appreciable and i am proud to be an indian

</BODY>

</DOC>

Sample query TREC file with 2 queries:

<topics>

<top>

<num>1</num>

<title>

hyderabad to howrah kono train ki diyeche ba debe durgapur jete hobe any idea jodi train chare then timing gulo ektu help korben

</title>

<desc>

hyderabad to howrah kono train ki diyeche ba debe durgapur jete hobe any idea jodi train chare then timing gulo ektu help korben

</desc>

<narr>

hyderabad to howrah kono train ki diyeche ba debe durgapur jete hobe any idea jodi train chare then timing gulo ektu help korben

</narr>

</top>

```
<top>
<num>2</num>
<title>
goto one week dhore amder balcony te paira der utpat khub berece nongra krce khub
aar anek jaigai phone o korlam paira der net laganor jnno kintu kono lubh holo na
noi keo phone dhore na noi ashbo bol a ashena can you guys help us out kono number
bah address provide korte paren jara kondapur a ashe balcony te net lagea debe
</title>
<desc>
goto one week dhore amder balcony te paira der utpat khub berece nongra krce khub
aar anek jaigai phone o korlam paira der net laganor jnno kintu kono lubh holo na
noi keo phone dhore na noi ashbo bol a ashena can you guys help us out kono number
bah address provide korte paren jara kondapur a ashe balcony te net lagea debe
</desc>
<narr>
goto one week dhore amder balcony te paira der utpat khub berece nongra krce khub
aar anek jaigai phone o korlam paira der net laganor jnno kintu kono lubh holo na
noi keo phone dhore na noi ashbo bol a ashena can you guys help us out kono number
bah address provide korte paren jara kondapur a ashe balcony te net lagea debe
</narr>
</top>
</topics>
```

4.5.3 Making Query Relevance (QRel) File

As a first step, we convert our documents and queries to the required TREC format. Then we run the Information Retrieval pipeline consisting of setup, indexing and batch retrieval using the Terrier IR platform. For the batch retrieval, we use pre defined famous models provided by Terrier, namely, **BB2, BM25, DFR_BM25, DLH, DLH13, DPH, DFRee, Hiemstra_LM, IFB2, In_expB2, In_expC2, InL2, LemurTF_IDF, LGD, PL2 and TF_IDF.**

We then take the top 100 relevant documents given by each of these models and take a union of all these documents to get the top relevant documents for each given query. We took these relevant documents and manually marked each document with True or False corresponding to whether each document is actually relevant to the given query. This task was done manually by two different persons and then checked whether their relevance judgements matched and a score was generated.

4.5.4 Process Flowchart for Retrieval

The document collection, preprocessing and indexing is a common step that is followed. In the first part, the queries are processed and passed through a query expansion step using any phonetic matching algorithm and then passed through the information retrieval module, as shown in figure 4.11.

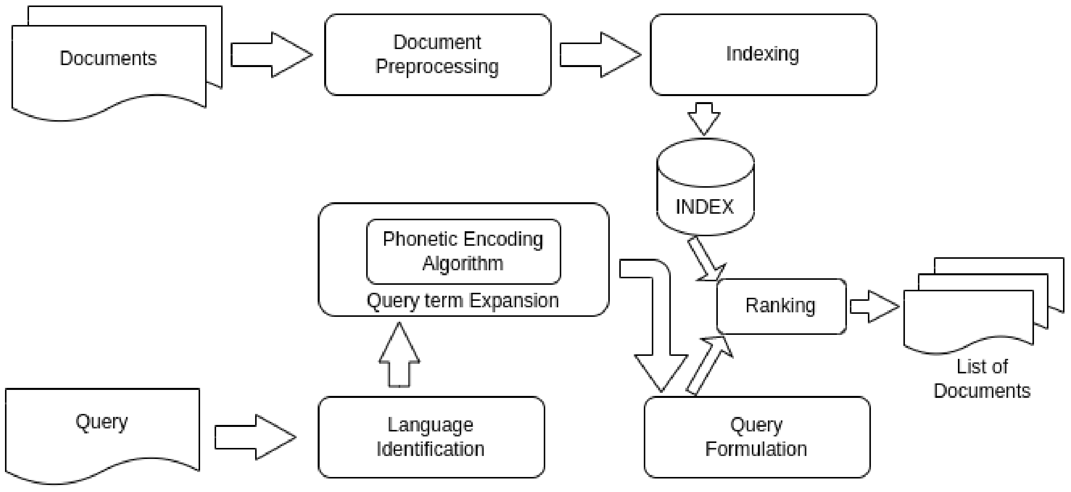


Figure 4.11: Process flow for retrieval when we use Soundex, Phonix and Hindex for all Query terms

In the next part, everything remains same except the language of the query terms is identified using the module discussed earlier and only the name entities, Bengali and Hindi terms are passed through the query expansion step using several combinations, which will be discussed later, and all other terms are kept intact, as shown in figure 4.12.

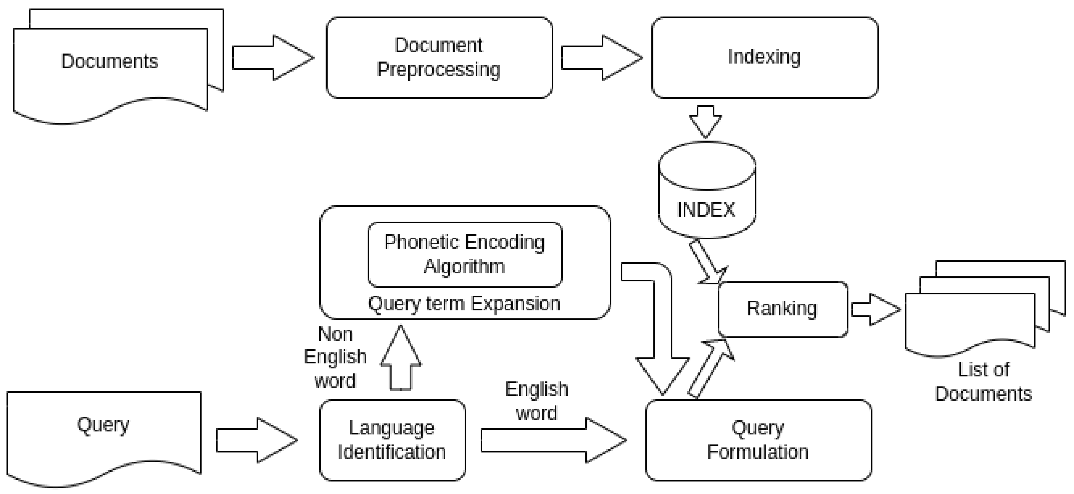


Figure 4.12: Process flow for retrieval when we use Soundex, Phonix for Name Entity terms, Hindex for Bengali and Hindi terms and keep English term as it is

4.5.5 Query Expansion [1]

Query expansion, or QE (as it is known in the text retrieval field), is a technique for improving search precision. The main concept is to utilise the results of the original query to reformulate the question and execute a second-pass search to acquire more precise results. Given its inherent attraction to user interactive systems and application to large-scale search, there have lately been a few attempts from the standpoint of vision to apply this notion to large-scale picture search. The procedure entails choosing and adding phrases to the user's query in order to minimise query-document mismatch and hence improve retrieval performance.

This study focuses on phonetic encoding for transliterated Bengali words and how it might be utilised for retrieval. This encoding assists in the recognition of phonetically similar phrases, which is then employed in query expansion to increase retrieval performance.

Thus to expand a query, we take a term T and convert it into a code C_t given by any phonetic matching algorithm. Then we move through the corpus and find terms W such that its phonetic code $W_t = C_t$ and the edit distance between T and W is below some threshold and we take that term into our query and thus the queries get expanded.

The general approach for extracting the identical terms goes like this: for a given query q with n terms say q_1, q_2, q_3, ... q_n a system extracts all the phonetically identical terms based on a phonetic algorithm (Soundex, Hindex etc.) from the lexicon L for each query term q_i.

Following steps are followed for extracting identical terms.

- Once codes are assigned to a query term, all the terms having identical codes from the lexicon L are extracted for further evaluation.
- For the terms having the same codes, edit distance is computed using Levenstein algorithm from the query term (assuming every edit operation takes same cost 1).
- Finally, to select the query candidates threshold for edit distance is set to 2 (i.e. terms with threshold value 2 are selected for query expansion).

To improve the quality of terms better, for Hindex we use a bit different approach. While calculating the edit distance between two words, we use a weighted Hindex cost while replacing a character with other characters. All the characters in string are not equally important in transliterated text when there does not exist

```
ALGORITHM 1: Query term expansion (Hindex): $P.Hindex(term)$
Data: Query terms $q_i \in q(q_1, q_2, q_3, \cdots, q_n)$ and lexicon $\mathcal{L}$
Result: Returns set of qualified query candidate terms $q_c$
Initialization, $i := 1, j := 1$;
while ($i \leq n$) do
    Read query term $q_i$;
    $m = q_i.length()$;
    $P\_Code_{q_i} = P.algo(q_i)$;
    while ($!eof(\mathcal{L})$) do
        read current term $L_j$;
        $n := L_j.length()$;
        $P\_Code_{L_j} \leftarrow P.algo(L_j)$;
        if $P\_Code_{q_i} = P\_Code_{L_j}$ then
            $\mathcal{T} \leftarrow LevEdit(L_j, q_i)$;
            if $\mathcal{T} \leq 2$ then
                $q_c \leftarrow L_j$;
            end
        end
    end
end
```

Figure 4.13: Query Expansion Algorithm

any standard spelling rule. Characters that produce consonant sound have greater importance than that producing vowel sound.

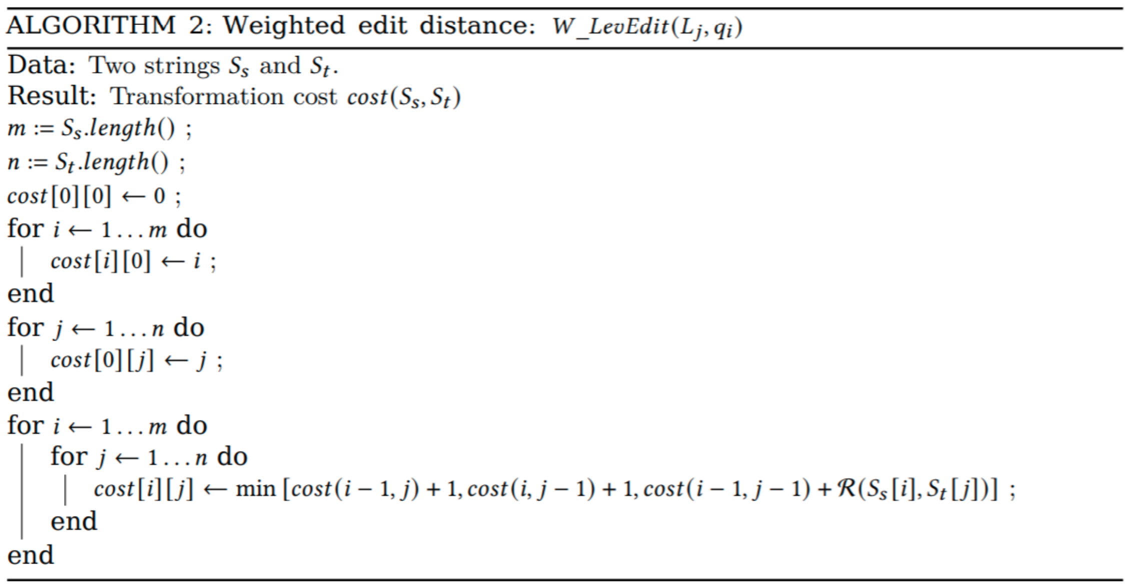
Steps:

- Follow all the steps of obtaining Hindex codes for each lexicon terms and extract those whose phonetic code match with that of query term q_i as candidate query terms.
- Use weighted edited distance for the terms whose phonetic codes do not exactly match with a given query term q_i
- Select all terms (S_s) with string length > 4
- For each such term, check whether its first character comes under the same phonetic group as the first character of q_i
- Select a term if edit distance between the phonetic codes of the term and $q_i <= 1$

- Compute distance between selected term and q_i and qualify the terms having edit distance <= 1
- For each selected term S_s and query term q_k calculate weighted edit distance $cost(S_s, q_k)$ according to Algorithm 2 where $R(S_s[i], S_t[j]$ is defined as

$$R(S_s[i], S_t[j]) = \begin{cases} 0 & \text{if } S_s[i] == S_t[j] \\ 0.5 & \text{if } S_s[i] \& S_t[j] \text{are non identical vowels} \\ 2 & \text{if } S_s[i] \& S_t[j] \text{are consonants from different phonetic groups} \\ 1 & \text{otherwise} \end{cases} \tag{4.8}$$

Thus we find words from the document collection that are phonetically similar to the query term using the weighted distance formula.



```
ALGORITHM 2: Weighted edit distance: $W\_LevEdit(L_j, q_i)$
Data: Two strings $S_s$ and $S_t$.
Result: Transformation cost $cost(S_s, S_t)$
$m := S_s.length()$ ;
$n := S_t.length()$ ;
$cost[0][0] \leftarrow 0$ ;
for $i \leftarrow 1 \dots m$ do
    $cost[i][0] \leftarrow i$ ;
end
for $j \leftarrow 1 \dots n$ do
    $cost[0][j] \leftarrow j$ ;
end
for $i \leftarrow 1 \dots m$ do
    for $j \leftarrow 1 \dots n$ do
        $cost[i][j] \leftarrow \min[cost(i-1, j) + 1, cost(i, j-1) + 1, cost(i-1, j-1) + \mathcal{R}(S_s[i], S_t[j])]$ ;
    end
end
```

Figure 4.14: Weighted Edit Distance for Hindex Code

Then for our further experiments, we take a hybrid approach where we identify the language of the terms in query first and then for Name Entities, Bengali and

ALGORITHM 3: Query term expansion with weighted edit distance (Weighted Hindex)

```
Data: Query $q$ and lexicon $\mathcal{L}$
Result: Returns a set of qualified query candidate terms $q_c$
Initialization, $i := 1$;
while $q$ not empty do
    Read query term $q_i$;
    $m := q_i.length()$;
    $P\_Code_{q_i} \leftarrow P.Hindex(q_i)$;
    Initialization, $j := 1$;
    while $(!eof\ (\mathcal{L}))$ do
        Read term $L_j$;
        $n := L_j.length()$;
        $P\_Code_{L_j} \leftarrow P.Hindex(L_j)$;
        if $P\_Code_{q_i} = P\_Code_{L_j}$ then
            $\mathcal{T} \leftarrow LevEdit(L_j, q_i)$;
            if $\mathcal{T} \leq 2$ then
                $q_c \leftarrow L_j$;
            end
        else
            if $(|L_j| > 4)$ & $\left(P\_Code_{q_i}[0] = P\_Code_{L_j}[0]\right)$ & $LevEdit(P\_Code_{q_i}, P\_Code_{L_j}) \leq$
             2 & $(Weighted\ Hindex\ cost \leq 1)$ then
                ‘ $q_c \leftarrow L_j$;
            end
        end
        $j := j + 1$;
    end
    $i := i + 1$;
end
```

Figure 4.15: Query Expansion Algorithm using Hindex

Hindi terms (with ‘en’, ‘bn’ and ‘hi’ tags), we use combinations of Hindex, Soundex and Phonix.

The results of all the Query Expansion techniques discussed above are shown below for a sample query.

Original Query

“hyderabad to howrah kono train ki diyeche ba debe durgapur jete hobe any idea jodi train chare then timing gulo ektu help korben”

English Translation:
"Is there any new train from Hyderabad to Howrah? Need to go to Durgapur. If anyone has idea about the timings, please help."

Soundex for all terms

"hyderbad hyderabaad hydrabad haydrabad hyderabadi hyderaabad hydrabaad hyderabad hyedrabad t2 0th 4ta twd ti teeo tuyo tow 1ta teo thy 6t 1to tao the 2t tui tw tooo thu tta toa ty 1ti teyo taao toy tht 2ta tha toho 2toi tou toh taoi ttoh tee te taou 7th 2toh 9ta tho tia thi 2toy tye 7ta 5te 3ta 8ta 4th teh tey tei tea 8th two tya tdo tau tu thw 4te toea th toe 5th t20 t9 tyoe thoo tay tue tt tuo too tai 3te tooh 0to tie tiu 6th to ta tuh 9th 2tow thou tto taa t 7t tae taw taio tuio 2to tah 5ta tua 6ta ttoi toi theo teu howrah horaha howar howhrah hora konno knoo kono kena kine khon kkno kene kin6o khoni kina kini kmn konoha konow kwon knu khna konoi konu kenoi kan0 kaano kne kohno kin2 kn konay kkhno kone keni kann kon kmne khano kuno kan kuna knooo khomo konoei kyno kane kjne khone kcn keno knoe konya know knw kun kinao konne kana khona kmmo khono kunu khn kanu kyano koma koini kenao kemo komao kano kani kin kjon kkn kome konui k9n kona kno kino kom komu kommo kon0 ken koni kaon trainee turan trian trun trn trane traim teran taran turin tran traan tron trina tram train kui kia kh k3 8k khoi koh keui kaa koyi 7k kiyu khie ki koei kja kc ko kau koe 2k ku ke kw 4kg keo kiu kj k4 ki6u k20 koi kchi k kou kao kiwi koy 2kg kew kihu kow 8k3 key k21 kahi k6e 7kg kyo 8kg 4k ky k6 kai 5k ka kie k6u khai ka6 kwa 5kg kgs kohi kee kiii kei khu ki6i ku6 kse kwi kae ks k11 koo khe keu kii kiye kyi 6k kih kha ka6i kaw kss kay kg kq k2 keh kyu 3k keoi kk kieh kho kiya kiei 1k ksj kya 9k diache diywchi dicche deyechi diyechhe dieoche diyechhi diyechi diechei diacche diyachi dioache diychi diche diyechiu diyecjo diyacha diyechho diyecho diyecche diechi dichche dieyechi diechhe dieche diyeche diecho diiche baaa beah b6 bbah bf bou bi bbu bho bahh bp 9b buy bua bea bo3 bewa boi baa bih bfb bawa bvb b3 by byea bio 5b bie boo bbw bey 1by bo b10 bahi bb 2b biya b2 b baah be bpf bba bei bawi bahu baho boe ba bhu boy bah biw bow be9 8b bay bhai bw bff biy bby b52 bye bhao bowa bh bee b8 b0 bae bu bayu b4u 1be b12 baha bv bhi bhau bai baea bhay bwa debi dep debey debeo dubie dewabe dubu dubey dbo dabi dub deboo debe1 dobe deboi def dwbe dubbe debei dope deb dab debu dave deva dobo dbe devi diba daba dove dba dubb dev dob debbie depp dhobe dfb db dub6e defy dive deep debo dbee dibe dibi doba dibo dubei debe dube durgapuja durgapur durgapujar durgapujo jeteo jetoh jat jetei juto jtei jot jet jte jetai jata jeta jedi jty jt6e jote jite jut jethu jutte jeto jto jed jta jete juti jiite jeeta jayte jste jetou jude jeet joto jita juta jetow jaite jate jst jath jetoi jetha jute jati joti jota jit jetae jiteo jetay jitey jato jaate jitte hope hive hbo hbee hop hobby hbeo hibe hoof hope5 hobo hub hobbee hobei hoyba hobay hba hb hobeei hbei hb3 hob hobe3 hopf heby hbby habu hbbi hobe hobu hby hobeo hebi hoobe hhobe have habi hobe99 hab hbbf haba hobi hbe habei hbp habe hbi hobee hobey hobe6 hobw hoba hype hoibo any an aona aneo ame ana anao anyo ane amn ammy annu aano 6ani aina ania anai amiy an8 aniye amne amm 6ana ain any15 anyy ayna aane anu anni an6o am aoni aan ani amy ama an6e aim anya ainu aam amay ami anyay anno awm aana anne ann anna an6i amo annyo ano amni anoy aini id iota idea ide ideaoi ida ita jodie jtei joat jot jadu jidi jyda jtoi jhadi jedi jote jdi jotoi jyoti jdu joyti judi jed joday juti jd jude joto jadi jodio jati joti jota jodi jdio trainee turan trian trun trn trane traim teran taran turin tran traan tron trina tram train charo chare chur cher chary chiri chaira carr chaire chorie care chore chara chari chaaro chaar chra charao charr chhire chhare chhere churi chaare chehre charia chero chiru chor chir cherei cara chire chri chharo charai cchara chhure choar chair chkari chayer chhar char6i chariye chora chhara chure chere chori char ciara charie car core chura charu chiro choro chera cure teen teno tan tene tena tthhn than thani then tmn ten tme tun tahan teni theme tain tohn thonu temn taan toin thn tuhin thena tian tem ton 7then tn teem town thane ten2 them thin thana tham tin tien tyan twin thom tuan timeing tuning timings timing timimg tamang gaalo gol guloo gulou gole gle golo galio gli gul goli gal gyalo gaylo ghule gulao guloe gala galeo gulie gola gela g6ilo gel guli gulu gailo gulia galo gilo geleo gili gale gila guolo galoo galoi gula gully gile gulow gl gele guloye gulo golu geli gule ge6lo guloy guloi glow gelo gulio gil glo g6lo guall gali galao gill ect ektu ekhte eakta ekti ektai est exta ecta ext4 ektuo esti ektay esta ectu esto ektaa ektao ekota ekto eektu ekta ext ekte este ekt halep help helpi holf hlp half helop helpo korbona korbena koraben1 korbeuni karben krbena korben korban krben korbina koraben karbon korbenna korbaen”

Phonix for all terms

“hydrabaad haydrabad hyderaabad hyderabaad hyderabadi hydrabad hyderabad hye-drabad 3te toi 4te tea tour tia teeo tee two tojo tau 7ta tai 5ta 2toy thi tay tsi tah ty 4ta toe tei the toho dt 0th tha tae 8th tuyo teu to 5th pti thou ter tao teo 5te 8ta tw tore taa toa 6ta 4th too 2toi 1to teh 9th ti 6th toy tooh tiu 0to toea 2toh toh tuh thor tie ta pta teyo thw tujo tor taio 1ti tar taoi th tue tre taor 3ta 2to tuio 9ta tya tow thoo te theo toja tho tua 1ta thy tu 2ta thu tou tsu 2tow tyoe taw tui tey 7th tye 2tor tooo tuo taou tir tur horaha hora howrah howhrah kano kin2 konoei konow kona khomo como kaano kon0 koma kmne keno khna kaon korno kinor kin6o kyano kome konne komor kono kana kunu kon kohno konoha keni komu kom kene khona kann khono konar komao korne konu kina kone koni khn kan kane kemo kmn kenao khon kan0 kuna kin ken kwon konya kanu kino kun cone kena con kini khano khoni kommo kuno khone korna koini konui kmmo konay kinao kani conn kenoi konoi kine kojon kyno kormo konno trn tran taran turan trun trina tram traim tron teran trainer trane turin trian trainee train traan khie cu keh kh kha kaa kai kow kiu qu kao kou kor ke kiya k6e keji khir kwa co kay kaw kiwi kie keo ky kee koei kire koy khu keu kyo q kau keui cai ker kihu kyu kew kahi kar kho ki6i ko ka6i kia kii ku6 khe key kih ca ku kije kiyu kiei kiye khoi ka6 ka kw ki6u kwi khai koh ki kui kyi kieh koo koe kre koyi kei koi kya kae qui kohi k6u kiii keoi kir diyechhi diyecjo diyecho dioache diiche dicche diyechhe dieche diyechiu diyecche diche diecho diyechho diyechi dieyechi diyacha diechhe diyeche diechei diyachi diywchi dieoche deyechi diache diacche diechi diychi bhau baah boja baje boi bayu bur bor baji 3bar bu bow baha 7bar 1by bay 1bar bare be9 bhi boar baho bh boe biy barr bewa bio bai bye bair bwa bea bw ber bei bowa bar bawi bawa 1be bhu bih baju biya byar buy bou bhay bagh bae bua bah bo3 b4u ba bahu be by bahh boy byea bhao bo bhar 2bar bey bi bhai beah bir baea 4bar baja bho boo baa buja biw bie bear baar bahi bee dob dwbe dabi dubbe dubb dewabe dub dhobe deboi debeo dab debei derby db derp dep debbie debe doba daba debor debo dibi dubei dba dbe dobo dbee dube dubey debe1 debey dubie dobe depp deb deboo deep debir dub6e debu dibe dbo debi diba dibo debar dope deber dubu durgapuja durgapur durgapujo durgapujar jutte jute jote jaite jeeta jetai jurte jitte jetei jeto joto jata jut jita jat jto jati jetow jeta jeteo jte jiteo jetae jed jetoi jetay jetou jty jeet jayte juti jeder jite jaate jetoh jato jedi jet jetha jethu jta jiite jude jtei jate jota jath jete juto joti juta jot jit jt6e jitey jetar hobi hobbee hoyba hobo hobe3 hub hb hober hob hba hebi hoobe hab hbo habu hope5 hb3 hbei hbe hby hobeei hoibo hobey hobay hobei hobee hobu hobw hope harbe habi haba hype hobby hobe99 hbp hobar hobe hbi hobe6 hhobe habe hoba hbee hop hibe habei heby hobeo hbby hbbi hbeo hber ain arm aniye ami in aneo ann amiy anu 6ana enh aoni anar onyo an6o onn anyay ina anary amm 6ani any une amne anno iniy aini ammy amy ainu inu anne arno ena anni aano anao ono anyo un amb anyy an6i army aina amni annyo en ania ene ian yana ama oni ano aona ana on ony onu eni aan anya aim ini anna ane amay am eno amo in2 uni on9 ona ani any15 ayna an8 uno aam anoy aana anyer ame onay inh onny una anai an6e onw awm an inn amn inm aane one erny annu id eda ader idhar oder adha idea ude ode odda ida iota ita udta adeo odia ide adda aide eder irda ideaoi adeu ede edey ada idur idar eida jyoti jote joto joday jati jed juti jodi jadi jyda jadu jidi jdi joyti jodie jedi jdu joat jotoi jodio jorda jude jtoi jordi jhadi jtei jota jdio jot judi jd joti trn tran taran turan trun trina tram traim tron teran trainer trane turin trian trainee train traan charr chary chayei chao chaoo char chhe chie chye chere chiye chore chure chhere chaho chaare chaye chire choar chaay chehre chir che chawa chare chur chhire cher chor chaii chaa chayer chee chae chay chhure chaoa chaya chaar cheye chai6e chau chuye chaiye chhar chohe chhare choe cha66e chahe chai chair chayo chaire cha ten2 torn teni thani town tem tahan tain tian theme turn thana taan then tuhin teem thonu tn teno tene tme thena tham 7then tun than tan thane temn thn toin them thom thin tien tmn tuan twin tyan tena tin ten teen ton tohn tuning timeing timimg timing timings tamang galoi goli gil guli golo gili gulie gaalo gali gile golu gele gel gulio ghurlo guall gill galio guulor g6lo ge6lo guloi gala gailo gila geli gulou guloy gl gulir guloe galeo gular gyalo gelo gela gol gaylo guloo gula galo glow gulao geleo gule gul ghule galoo gilo gulia gole gale gulo gli gulow gola galao gully gulor glo guolo gal guloye g6ilo gle gulu ekota ekta aktuo ekt ectu ektaa aekta ekhte ektur akte ektuo ektai akto eakta akdu aktu akti ekte akt aktur ektao ukti okte ikto ukte ektar iktu ekto ecta ukto akta eektu ect ekti ektu ektay helpi help helop halep hlp helpo helper korbona korbenna koben korbeuni korbina korban korbena karben korbaen korben khaben karbon”

Hindex for all terms

“hyderabad hyderabaad hyderaabad toa tot to tua toto tuo tto howhrah howrah kokono kokn kono konno kon kokon kokkno train trained trainer kee kiki ki kik kiii kii diyechi diyeche vaa baba bab ba vba vaba wa va wav bba baaa vab bawa bava baa bwa debr debei dewer debeb dev dew debe deb deber durgapur jetae jet jete jetei jetai jaite hobei hobe hhobe anya anyy anyay any ide idea zodi jordi jori jodi train trained trainer chare chade charche chaare charai chader then timing gulo ektuku ektu ektuk help korben korbaen”

Hindex for Bengali/Hindi terms, Soundex for rest

“hydrabaad haydrabad hyderaabad hyedrabad hydrabad hyderbad hyderabad hyderabaad hyderabadi toa toto tot tto tua tuo to hora howhrah horaha howrah howar kokkno kokn kono kokon kokono kon konno trina turin tran tram traim turan trian trane train trun traan trn tron teran taran trainee kiii kii ki kiki kee kik diyeche diyechi ba bwa vba bab bava va vab baa bba baba wav bawa baaa vaba vaa wa debeb debe dew dev debr deb debei dewer deber durgapur durgapuja durgapujo durgapujar jet jete jetae jaite jetai jetei hhobe hobe hobei amni an8 anyo ammy amy anyy anai ann ane ania aan aniye anao amm 6ani aano an ame annyo any an6e an6i ana aona amay any15 ainu anyay amn aina ama ami anoy anya amne annu ain amo aam aoni aim anne aneo anni ano aini awm amiy aane ani anno aana am anna ayna anu 6ana an6o ida id ideaoi ide idea ita iota zodi jori jodi jordi trina turin tran tram traim turan trian trane train trun traan trn tron teran taran trainee charche chare chaare chade chader charai 7then tohn tain tthhn tan twin then tene taan toin thani ten town them teni thane thn thom tena ten2 tun tme thena tn tuhin tmn teem tian temn tem thin tham thonu theme tyan teen teno tahan thana ton tin than tien tuan timeing timings tamang timimg timing tuning gulo ektu ektuk ektuku helop half helpo helpi hlp holf halep help korbaen korben”

Hindex for Bengali/Hindi terms, Phonix for rest

“hydrabad hyderaabad hyderabaad hyderabad hyderabadi hyedrabad haydrabad hydrabaad to toto tua tto tot tuo toa hora howhrah horaha howrah kokkno kon kokn kono kokon konno kokono trane trn trainee trian trainer train trun trina taran tran traim teran tron tram turan turin traan ki kiii kii kiki kee kik diyeche diyechi bab baa vaa va baba vab bawa bwa vba wa bava baaa ba vaba wav bba dew debeb deber deb debr dewer debe debei dev durgapujo durgapuja durgapujar durgapur jetai jaite jetei jet jetae jete hobe hhobe hobei an6o anya ania ono ann amay anni anne un ami uni arno ana anu anai anao aoni ona aan yana ane any anoy en 6ani inu 6ana annu anary ina aim onn amm eno aana an8 army aini amne eni anyay ene one ony anar amy onyo erny ano enh aneo on9 anyy awm aona una aam anyer inn amni inm am annyo ama ani inh ini aane uno oni on anna amo ian amiy anyo aniye aano ena amb ain arm any15 anno in onny ame onay ainu an onu in2 amn iniy aina ayna an6i an6e une ammy onw eder irda id udta eida adeu oder ida adda adha iota idar aide ede idhar idur adeo ader ude idea ode odda eda ideaoi edey ide odia ada ita jordi zodi jori jodi trane trn trainee trian trainer train trun trina taran tran traim teran tron tram turan turin traan chare charche chade chader charai chaare thani teen turn teni taan tene tohn ten twin tun tian teem theme tien thana thane toin them then tn thena tin tan 7then than ten2 town tham thom tena tain tahan thonu temn tem tmn teno thin tuan thn tuhin tme ton torn tyan timings timimg timeing tuning timing tamang gulo ektu ektuku ektuk hlp help halep helpi helop helper helpo korben korbaen"

Hindex for bengali/hindi terms and name entities

"hyderabaad hyderaabad hyderabad tto tua toto to tot toa tuo howrah howhrah kokkno kon konno kokono kono kokon kokn train kik kiki kee kii ki kiii diyeche diyechi bava bawa vaa wav bba ba vba bwa va bab baba vaba wa vab baaa baa dev dew deb debeb deber debe debei dewer debr durgapur jetai jaite jetei jete jetae jet hhobe hobe hobei any idea jodi zodi jori jordi train charche chader chare charai chaare chade then timing gulo ektuk ektuku ektu help korbaen korben"

Hindex for bengali/hindi terms, Soundex for name entities

"hydrabaad hyderabaad hyderbad hyderabadi hydrabad hyderabad hyderaabad hye-drabad haydrabad tuo toa to tot tto tua toto howar howrah hora howhrah horaha konno kokono kokkno kono kon kokn kokon train kee kii kiki ki kiii kik diyechi diyeche ba vba vaba bwa bab va wav vab baba bawa baa bava baaa vaa bba wa dewer debei debr dew debe dev deb debeb deber durgapur durgapujar durgapujo durgapuja jet jetai jetei jetae jaite jete hobei hhobe hobe any idea jori jodi jordi zodi train chader chare chade charche charai chaare then timing gulo ektuku ektuk ektu help korbaen korben"

Hindex for bengali/hindi terms, Phonix for name entities

"hydrabad hyderaabad hydrabaad hyderabaad hyedrabad hyderabad hyderabadi haydrabad tuo toto tua toa to tto tot howrah hora howhrah horaha kono kokon kokono kokn kon kokkno konno train kee kiki kii ki kiii kik diyechi diyeche vba ba bab va wa wav vab bawa vaa vaba baba bava baaa bwa bba baa dewer dew deb debe debeb debei debr deber dev durgapuja durgapujar durgapur durgapujo jetai jet jetei jete jaite jetae hobe hhobe hobei any idea jodi jori zodi jordi train charai chaare charche chare chader chade then timing gulo ektuku ektu ektuk help korben korbaen"

4.5.6 Zero Shot

Pyterrier framework has been used to evaluate all the models on Google Colab's GPU runtime environment. Pandas dataframes are first constructed for the documents, queries and relevance judgements for the given codemixed dataset and the index is generated corresponding to the documents in the dataset with the text and docno as metadata which is required by the neural models. Porter's stemmer and stopwords removal have been used as preprocessing steps prior to creating the codemixed index. Below is the description of the different models used:

1. **KNRM:** For this neural reranker we have used weights that have been tuned on Med-Marco dataset and for word embeddings Fasttext has been used. In the experimentation phase we have added a KNRM reranker model in a pipeline after the BM25 model. We repeated the experiment for top "Cut" documents retrieved by the BM25 model, where Cut is varied from 10 to 100 as multiples

of 10.

2. **Vanilla Transformer:** This model utilizes a transformer model that has also been trained on the Med-Marco dataset and utilizes Scibert embeddings. In the experimentation phase we have added a vanilla transformer reranker model in a pipeline after the BM25 model. We repeated the experiment for top "Cut" documents retrieved by the BM25 model, where Cut is varied from 10 to 100 as multiples of 10.

3. **MonoT5:** Since MonoT5 is already pre pre-trained reranker, we have utilized it to rerank the top "Cut" documents retrieved by the BM25 model, where Cut is varied from 10 to 100 as multiples of 10.

4. **DeepCT:** This utilizes DeepCT model which has been trained on MS Marco dataset to index out codemixed documents and then we used these term weights to rank and evaluate the top 100 documents retrieved using the BM25 model on the DeepCT codemixed index.

4.5.7 Information Retrieval

As discussed earlier, Terrier IR Platform was used to load the corpus, build the indexes and retrieve meaningful documents using several State of the Art IR techniques. At first, the platform was set up as per our requirements. The given data and query were converted to the required trec format. Indexing was done to ease the process, followed by batch retrieval. Each query produced a number of indices corresponding to the given data, sorted in decreasing order of relevance.

In each turn, the documents were kept constant and the trec file for the queries were changed corresponding to the expanded queries for each of the phonetic matching algorithms discussed above. The original queries were placed inside the "<title>

</title>" tags and the expanded queries inside the "<desc> </desc>" tags. The final batch evaluation results were then recorded and compared to find out the phonetic matching algorithm that worked best with query expansion.

A sample query TREC file for Hindex QE with 2 queries is as follows: <topics>
<top>
<num>1</num>
<title>
hyderabad to howrah kono train ki diyeche ba debe durgapur jete hobe any idea jodi train chare then timing gulo ektu help korben
</title>
<desc>
hyderabad hyderabaad hyderaabad toa tot to tua toto tuo tto howhrah howrah kokono kokn kono konno kon kokon kokkno train trained trainer kee kiki ki kik kiii kii diyechi diyeche vaa baba bab ba vba vaba wa va wav bba baaa vab bawa bava baa bwa debr debei dewer debeb dev dew debe deb deber durgapur jetae jet jete jetei jetai jaite hobei hobe hhobe anya anyy anyay any ide idea zodi jordi jori jodi train trained trainer chare chade charche chaare charai chader then timing gulo ektuku ektu ektuk help korben korbaen
</desc>
<narr> </narr>
</top>
<top>
<num>2</num>
<title>
goto one week dhore amder balcony te paira der utpat khub berece nongra krce khub aar anek jaigai phone o korlam paira der net laganor jnno kintu kono lubh holo na

noi keo phone dhore na noi ashbo bol a ashena can you guys help us out kono number bah address provide korte paren jara kondapur a ashe balcony te net lagea debe

</title>

<desc>

got goat goto onei one onne beek week dhorre dhhore dhore dhorai dhorei amamder amder amader amaderr amaader aamader amded aamder amarder amade balcony tei tai tea tete tae taie tet te paida paira rer red de dead dear dai der dede dae ded dei dea rear re rere utpat khub khubbb khuub khuuub khoob khubb berece nongra nogra krce krche khub khubbb khuub khuuub khoob khubb adaar arr ad aaar ara aadar aar aada ar arrr aad adar ada add addar anekkk anek aneke anekei jaigai jaegae fone phonei phone o ua oa uo korlaam korlam paida paira rer red de dead dear dai der dede dae ded dei dea rear re rere neat nete net laganor jnnno jno jnno kkintu kikintu kinttu kintu kintuuu kokono kokn kono konno kon kokon kokkno lubh holo hol naaa naa nana nna naan na nan noi noe keu kiu keo fone phonei phone dhorre dhhore dhore dhorai dhorei naaa naa nana nna naan na nan noi noe ashbo aasbo asbo bol boll volo bolo bolbo bowl woll vol bollo a aa aaa ashen aseina aasena asenan ashena asen asena asenna cann can you guys help ush uss us oos aot out kokono kokn kono konno kon kokon kokkno number numbered numbed babah baha baah waha wah wahh baahh waah bawah bbah bah bahh adress addresa addressed address provided provide provider kortre kortei korte kortai parena paarena paaren paren jaar jadr zada jaara jara zara jar jjara zaara kondapur a aa aaa ashai ashei aashe asai ashe aase asses ase ashes asei asse balcony tei tai tea tete tae taie tet te neat nete net lagai laglei lagega lagea lagei laage lagae lage lagle debr debei dewer debeb dev dew debe deb deber

</desc>

<narr> </narr>

</top>

</topics>

4.6 Results

4.6.1 Zero Shot

- **Base Result:**

S.No.	name	map	recip_rank	ndcg	ndcg_cut.10	mrt
1	BM25	0.319334	0.632875	0.573167	0.394532	81.780515

- **Neural Rerankers:**

S.No.	name	map	recip_rank	ndcg	ndcg_cut.10	mrt
cut = 10						
1	BM25 » KNRM	0.245757	0.609568	0.365754	0.385240	91.312363
2	BM25 » vbert	0.218396	0.532216	0.338823	0.357550	249.849831
3	BM25 » MonoT5	0.231972	0.567718	0.350924	0.369455	389.734604
cut = 20						
1	BM25 » KNRM	0.260557	0.618592	0.407359	0.374655	96.974283
2	BM25 » vbert	0.235030	0.511391	0.376557	0.328779	395.615256
3	BM25 » MonoT5	0.246694	0.554413	0.387767	0.343720	666.700302
cut = 30						
1	BM25 » KNRM	0.255028	0.567168	0.413569	0.358016	100.524321
2	BM25 » vbert	0.237718	0.517602	0.392720	0.324987	544.966424
3	BM25 » MonoT5	0.253310	0.551617	0.406219	0.325876	950.088736
cut = 40						
1	BM25 » KNRM	0.275274	0.588590	0.446541	0.373333	103.920086
2	BM25 » vbert	0.239675	0.526459	0.410591	0.324156	691.766516
3	BM25 » MonoT5	0.254197	0.529999	0.419982	0.330472	1224.181060
cut = 50						
1	BM25 » KNRM	0.267323	0.571596	0.448028	0.352353	109.547629
2	BM25 » vbert	0.237704	0.528762	0.417985	0.314837	840.560879
3	BM25 » MonoT5	0.255282	0.512938	0.428101	0.329526	1512.674665

cut = 60						
S.No.	name	map	recip_rank	ndcg	ndcg_cut.10	mrt
1	BM25 » KNRM	0.262628	0.562678	0.452864	0.343002	119.458731
2	BM25 » vbert	0.241534	0.525958	0.428256	0.312345	982.857105
3	BM25 » MonoT5	0.256186	0.525847	0.440276	0.334602	1790.537858
cut = 70						
1	BM25 » KNRM	0.248695	0.546822	0.448307	0.324247	114.868405
2	BM25 » vbert	0.241534	0.525958	0.428256	0.312345	982.857105
3	BM25 » MonoT5	0.263074	0.518859	0.449663	0.340973	2072.491896
cut = 80						
1	BM25 » KNRM	0.246063	0.538293	0.450578	0.326392	118.776692
2	BM25 » vbert	0.236545	0.514803	0.435412	0.302032	1272.853488
3	BM25 » MonoT5	0.258594	0.527406	0.452748	0.333215	2338.327154
cut = 90						
1	BM25 » KNRM	0.234713	0.528377	0.442303	0.313370	123.600964
2	BM25 » vbert	0.231225	0.514334	0.432208	0.295829	1418.763112
3	BM25 » MonoT5	0.255750	0.501140	0.447450	0.329188	2625.367774
cut = 100						
S.No.	name	map	recip_rank	ndcg	ndcg_cut.10	mrt
1	BM25 » KNRM	0.237978	0.526219	0.450951	0.317864	131.243698
2	BM25 » vbert	0.230331	0.498282	0.435886	0.291609	1553.618822
3	BM25 » MonoT5	0.258115	0.523597	0.458669	0.329761	2899.497418
4	DeepCT BM25	0.398713	0.723895	0.599657	0.489438	53.79597

4.6.2 Phonetic Encoding Algorithms

Here are the results after experimenting with phonetic algorithms on query expansion.

Phonetic Encoding Scheme	IR Models	MAP	RPre	P@10
Not applied	BM25	0.2214	0.2313	0.1920
	TF_IDF	0.2145	0.2194	0.1900
	BB2	0.2512	0.2567	0.2220
	InL2	0.2306	0.2402	0.1980
	DPH	0.2154	0.2223	0.1880
	LM	0.3390	0.3203	0.2800
Phonix(all)	BM25	0.0673	0.0697	0.0760
	TF_IDF	0.0665	0.0688	0.0740
	BB2	0.0533	0.0549	0.0580
	InL2	0.0703	0.0762	0.0760
	DPH	0.0280	0.0274	0.0260
	LM	0.0231	0.0153	0.0300
Soundex(all)	BM25	0.1235	0.1278	0.1220
	TF_IDF	0.1211	0.1240	0.1260
	BB2	0.0906	0.1013	0.1040
	InL2	0.1303	0.1342	0.1340
	DPH	0.0603	0.0536	0.0580
	LM	0.0472	0.0653	0.0540
Hindex(all)	BM25	0.2085	0.2237	0.1840
	TF_IDF	0.1989	0.2084	0.1740
	BB2	0.2057	0.2286	0.1880
	InL2	0.2097	0.2241	0.1940
	DPH	0.1855	0.1766	0.1680
	LM	0.2298	0.2487	0.1980
Hindex(bn/hi) + Phonix(rest)	BM25	0.1767	0.1948	0.1600
	TF_IDF	0.1715	0.1757	0.1580
	BB2	0.1539	0.1731	0.1640
	InL2	0.1816	0.2035	0.1740
	DPH	0.1161	0.1224	0.1160
	LM	0.0984	0.1183	0.1000
Hindex(bn/hi) + Soundex(rest)	BM25	0.2003	0.2115	0.1800
	TF_IDF	0.1905	0.1949	0.1740
	BB2	0.1934	0.2198	0.1900
	InL2	0.1990	0.2154	0.1860
	DPH	0.1552	0.1430	0.1400
	LM	0.1252	0.1398	0.1260
Hindex(bn/hi/ne)	BM25	0.2290	0.2400	0.2020
	TF_IDF	0.2212	0.2200	0.1980
	BB2	0.2389	0.2531	0.2120
	InL2	0.2322	0.2352	0.2080
	DPH	0.1998	0.1972	0.1660
	LM	0.2686	0.2749	0.2280
Hindex(bn/hi) + Phonix(ne)	BM25	0.2335	0.2427	0.2020
	TF_IDF	0.2277	0.2312	0.2020
	BB2	0.2440	0.2621	0.2140
	InL2	0.2386	0.2435	0.2100
	DPH	0.1988	0.1977	0.1720
	LM	0.2717	0.2788	0.2300
Hindex(bn/hi) + Soundex(ne)	BM25	0.2322	0.2440	0.2060
	TF_IDF	0.2248	0.2265	0.2060
	BB2	0.2412	0.2561	0.2180
	InL2	0.2361	0.2389	0.2140
	DPH	0.2008	0.2010	0.1720
	LM	0.2742	0.2822	0.2320

Table 4.7: MAP, RPrec and Precision@10 score on uni-Word Indexing

4.7 Analysis and Conclusion

Extensive experiments are conducted to test the proposed phonetic algorithms vis-a-vis other existing counterparts using the framework of an application domain, namely, CMIR. The objective is to see how the issue of spelling variations in the query terms are addressed during the search of Code mixed query and its comments(in Roman script). The retrieval performances on the use of some existing phonetic algorithms and a hybrid algorithm, that on the proposed one are tested with our own datasets.

Hindex extracts a moderate number of terms, that reduces the issues associated with too short or too long queries. Assigning separate weights to the consonant and vowel sound characters increases the candidate term counts for the expanded queries. The performance of different phonetic algorithms is also evaluated through a framework of IR setting: using data which we have collected from social media. Bengali-English code mixed data (Roman script) are retrieved for a given set of queries (50 queries) in Bengali-English code mixed (Roman script) text.

Also since generating the relevance judgement file (Qrel) is a manual and tedious job, adding all relevant documents from such a large dataset was a challenging job. We took some other alternative to generate that file, as discussed earlier. So there are cases where a relevant document is actually retrieved but not classified as relevant since it is not present in the Qrel file. Thus, the actual results in most cases are better in real life, even though they don't get reflected in the results table.

Chapter 5

Hate Speech and Offensive Content (HASOC) Identification in English, Indo-Aryan and Code-Mixed (English-Hindi) languages

5.1 Overview

Now, half of the world's population has internet access. People from all cultures and educational backgrounds interact on social media. Individuals and groups are stigmatised by communicating their thoughts and opinions through nasty and offensive rhetoric. User-generated content on social media, in particular, has become a breeding ground for derogatory language and hate speech. As a result, morale suffers, and mental suffering and trauma are unavoidable. As a result, information retrieval from social media data and the detection of potentially harmful languages are deemed critical. Almost all social networking sites include rules regarding abusive language, but finding them might be challenging. It is impossible to monitor the situation manually or with a fixed set of rules. Because hate speech and offensive language are natural languages, it is possible to employ natural language processing (NLP) methods to search for offensive material in textual data. The 2021 shared tasks on Hate Speech and Offensive Content Identification (HASOC) focuses on Indo-Aryan languages in three languages: English, Hindi, and Marathi. Subtask-1 and Subtask-2 are two subtasks of the shared tasks. Subtask-1 is divided into two sections: Subtask-1A, which is a coarse-grained binary classification, and Subtask-1B, which is a fine-grained binary classification. Subtask-2 is primarily concerned with identifying Conversational Hate-Speech in Code-Mixed Languages (ICHCL). The Subtasks-2 dataset includes tweets written in English, Hindi, and code-mixed Hindi. As a result, we were able to solve the linguistic challenges that come with social media posts. We employed publicly available pre-trained transformer-based neural network (BERT) models to tackle this problem, which may be fine-tuned for individual workloads. Furthermore, its multilingual functionality enables us to assess sentiment for comments including words and sentences in several languages. Both Subtasks, all three languages, and one Code-Mixed language were all completed by us.

5.2 Task Description

The goal of HASOC 2021 was to provide a testbed for systems that can automatically recognise hate speech and objectionable content from social media posts. In HASOC, there were three subtasks. They are discussed with examples in the below table 5.1.

- **Subtask 1:** Hate speech detection in monolingual text.
  - **Subtask A:** A binary classification task that classifies the sentiment of each tweet into one of the two categories described below:
    - **Non Hate-Offensive (NOT)** - There is no hate speech, profanity or objectionable content in the given post.

    - **Hate and Offensive (HOF)** - The post is hateful, offensive or vulgar.
  - **Subtask B:** Predicting the type of offensive post. This task is basically a multi-class classification. Once a post is classified as a HOF post, we need to predict the type of hate speech out of the following possibilities:
    - **Hate speech (HATE):** - This tag implies that the message is addressed to a group or a member of a group who is aware of his or her membership. Any harsh remarks directed against them because of their political ideas, sexual orientation, gender, social status, physical condition, or whatever else.
    - **Offensive (OFFN):** - This tag indicates that the post contains objectionable content, such as demeaning, insulting, or threatening an individual.
    - **Profane (PRFN):** - This tag indicates the presence of curse words.
- **Subtask 2:** : Identification of Conversational Hate-Speech in Hindi-English Code-Mixed Language given in the form of tweets.

5.3 Related Work

Over the last few years, there have been several studies on computational method to identify hate and offensive speech. Some prior works have studied blogs, micro-blogs, and social networks like twitter data [18], [19], [20], [21] and [22] as well as Facebook post and Wikipedia comments.

A couple of studies like [23], [24], [19] and [25] have been published where they focused on detecting whether a post contains hate speech or not, only two-way classification. Dinakar et al. [26] proposed an idea where they classify the posts based

Table 5.1: Example tweets from the HASOC2021 dataset for all classes

Language	**Sample tweet from the class**	**SubTask-1**	
		A	**B**
English	@Wari_gay Can't expect God to do all the work	NOT	NONE
	@bananapixelsuk That's why the whole thing is a load of crap. Corporate bollocks.	HOF	HATE
	@ndtv Shameless PM. What else can we say? #ShameOnModi #Resign_PM_Modi #ResignPMmodi	HOF	OFFN
	@UtdEIIis Really like how this list started with Dan Shitbag.	HOF	PRFN
Hindi	#किसानों_का_मोदी_को_धोबीपटका #ResignPMmodi https://t.co/nKTiocjjMl	NOT	NONE
	@anushka_s2 मूर्ख लड़की	HOF	OFFN
	सवाल यह नही कि वो मुझे वेश्या कहता है सवाल तो यह है कि मुझे वेश्या बनाया किसने ? -नवनीत	HOF	HATE
	धवन मदरचोद ज़िंदा है मर गया	HOF	PRFN
Marathi	तिने तोंड उघडलं ह्यांनी नाक दाबायला सुरुवात केली.	HOF	-
	आयुष्य खूप सोपं आहे आपण ते विनाकारण अवघड करुन ठेवतो...	NOT	-
Language	**Sample tweet from the class**	**SubTask-2**	
English-Hindi	@MovidMukt_India @srivatsayb Kaash tere sochne k hisab se duniya chalti	HOF	-
	@ashokepandit Sir, we will do kafi ninda only or book as per law?	NONE	-

on the frequency of offensive or socially non-acceptable words. Machine learning algorithms using TF-IDF characteristics are being utilised in social media to identify and categorise hate speech and offensive language [27].

Because of the scarcity of relevant corpora, the vast majority of studies on abusive language have focused on English data. However, a few research works have recently looked into abusive language detection in different languages. Mubarak et al. [28] deal with abusive language detection on Arabic social media, whereas Su et al. [29] offer a method for detecting and reverting profanity in Chinese. Hate speech and abusive language datasets for German and Slovene have recently been annotated by Ross et al. [30] and Fiser et al. [31] respectively, which paved the way for future work in languages other than English. Also many workshops have been organised to identify hate speech. The SemEval-2019 Task 6: Identifying and Categorizing Offensive

Language in Social Media (OffensEval 2019) [32] was the first competition towards detecting offensive language in social media (Twitter) only on English language. The SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020) [33] organised for the same proposes with four other languages Arabic, Danish, Greek, and Turkish. Germeval Task 2, 2019 [1] - Shared Task on the Identification of Offensive Language, Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC 2019) [2], (HASOC 2020) [3] try to identify Hate speech on English, Hindi and German language.

There have been some work exploring different aspects of offensive content like *abusive language* ([25], [28]), *cyber-aggression* [22], *cyber-bullying* [34] and *toxic comments or hate speech* ([23], [21], [24]).

5.4 Dataset

We have used the HASOC 2021 dataset which was taken from Twitter for a multilingual study including three languages: English, Hindi, Marathi, and one Code-Mix (English-Hindi). The table shows the corpus collection and class distribution.

5.5 Methodology

5.5.1 Preprocessing

The BERT-specific tokenizer breaks a sentence into tokens in a WordPiece-like manner during the primary preprocessing step. Hashtags and emoticons make up the majority of the data gathered from Twitter. In order to improve the performance we performed two twitter specific steps:

- We substituted all the emoticons with their actual text representations so that

[1]https://projects.fzai.h-da.de/iggsa/
[2]https://hasocfire.github.io/hasoc/2019/index.html
[3]https://hasocfire.github.io/hasoc/2020/index.html

Table 5.2: Statistical overview of the Training Data and Test Data for determining the final results

			Subtask-1A		Subtask-1B		
Data	Language	# Sentences	NOT	HOF	HATE	OFFN	PRFN
Train	English	3843	2501	1342	683	622	1196
	Hindi	4594	3161	1433	566	654	213
	Marathi	1874	1205	669	-	-	-
Test	English	1281	483	798	224	195	379
	Hindi	1532	1027	505	215	216	74
	Marathi	625	418	207	-	-	-
			Subtask-2				
Data	Language	# Sentences	NONE	HOF			
Train	Hindi-English	5740	2899	2841	-	-	-
Test	Hindi-English	1348	653	695	-	-	-

the model can see its actual meaning. For example, 😂 is replaced with "beaming face with smiling eyes". To carry out this preprocessing step we used a python module (**demoji**) which provides a convenient dictionary for emoji substitution.

- We replaced hashtags with their segmented constituents. For example, or example, "#IndiansDyingModiEnjoying" is segmented into"Indians", "Dying", "Modi", and "Enjoying". We carried out this process using the python module (**ekphrasis**).

5.5.2 Implementation

Every subtask may be thought of as a text classification problem. Our submission models were created by applying shared task data to a pre-trained language model. We chose BERT as our pre-trained language model because of its recent success and public availability in numerous languages. We used the bert-base-cased and bert-base-uncased models for both subtasks of the English language after completing preprocessing procedures. Furthermore, we submitted a run that was not preprocessed in any way. For both subtasks of the Hindi language, we used the bert-base-multilingual-cased model. For the Marathi subtask also, we use the same bert multilingual-cased model. BERT implementation provided by the pytorch-transformers library was used for the exprementation. As shown in the figure below, input is represented by a sequence of words. The subtokens are generated by adding a CLS token at the beginning and SEP tokens at the end. This sequence is then fed into the BERT model which produces an embedding for each word R_{W_i}, and a R_{CLS} corresponding to the CLS classification token. Transformers are used by BERT, which is an attention mechanism for learning contextual relationships between words in a text. A transformer uses two different mechanisms for its working in its basic form:

- **An encoder:** It reads the textual input.
- **A decoder:** It generates job predictions.

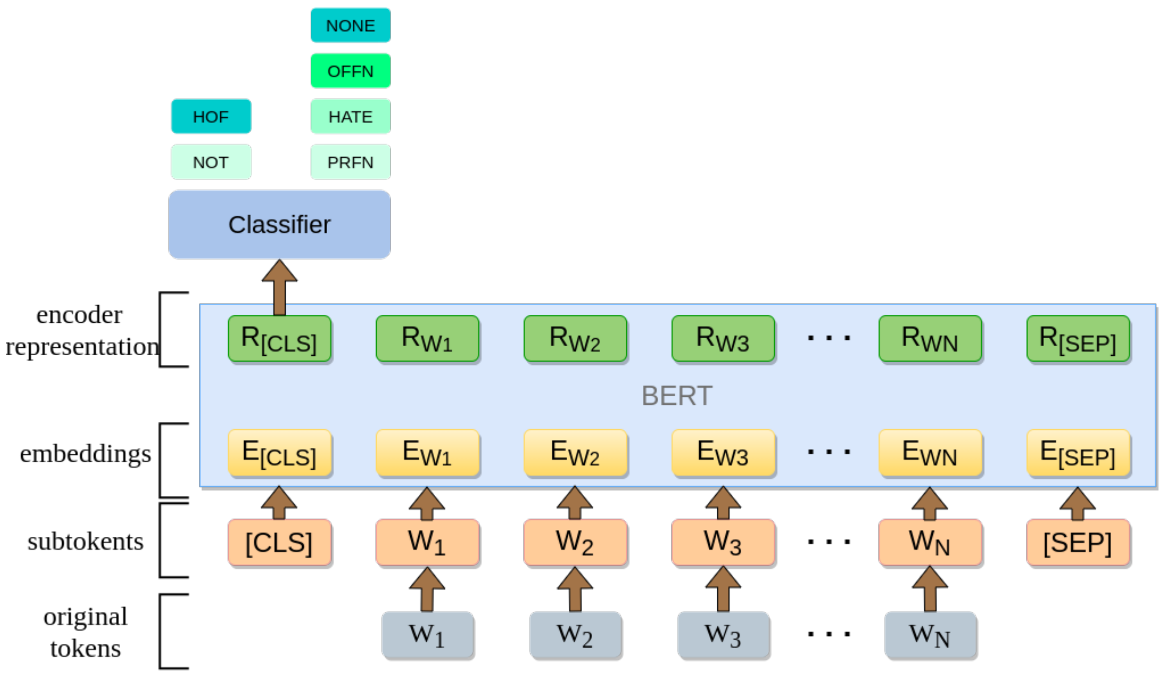


Figure 5.1: BERT model architecture for sequence classification

In BERT only the encoder step is required as BERT's main purpose is language model construction. The R_{CLS} vector is then used as a feature vector for a feed forward neural network with a softmax output layer to predict the probability distribution over the classes for the task at hand. Here the number of output neurons are defined by the number of possible classes which are 2 and 4 for subtasks A and B respectively. For the implementation, HuggingFace's transformers library was used. HuggingFace transformers is a Python package that offers pre-trained and customised transformer models for a variety of natural language processing tasks. It comprises BERT models that have been pre-trained and are multilingual, as well as alternative models that are better suited for downstream tasks. The implementation environment is the PyTorch library, which supports GPU computation. The BERT models were run using Google Colab. We trained our classifier for 15-20 epochs with a batch size of 32. The AdamW optimizer is used with a learning rate of 2e-5, and the dropout value is set at 0.1.

5.6 Results

We validated our model on the training and development sets since we lacked test labels. As our submission for each subtask, we chose the top models from each evaluation. Every System is evaluated using a Macro F1 score. The overall system's macro F1 score is the average of the different classes' F1-scores. Table below shows the best performing team and our official performances on the test data as shared by the organizers vis-a-vis the best performing team for all shared tasks of English, Hindi, Marathi, and code mixed Hindi-English language.

1. **English Sub Task A (submission 1):** BERT base multilingual cased (mBERT), 20 epochs without replacing emojis and hashtags, Maximum sequence length of 128 tokens, and batch size of 32.(Macro F1: 0.7579)

2. **English Sub Task A (submission 2):** mBERT, 20 epochs using emoji and

Table 5.3: Evaluation results on test data and rank list (Submission number in bracket)

Language	Subtask	Team Name	Macro F_1 score	Rank
English	1-A	NLP-CIC	0.8305	1 / 56
		IRLab@IITBHU (1)	0.7579	-
		IRLab@IITBHU (2)	0.7581	-
		IRLab@IITBHU (3)	0.7812	-
		IRLab@IITBHU (4)	0.7886	-
		IRLab@IITBHU (5)	**0.7976**	16 / 56
	1-B	NLP-CIC	0.6657	1 / 37
		IRLab@IITBHU (1)	**0.6093**	18 / 37
Hindi	1-A	t1	0.7825	1 / 34
		IRLab@IITBHU (1)	0.7471	-
		IRLab@IITBHU (2)	0.7440	-
		IRLab@IITBHU (3)	**0.7547**	13 / 34
	1-B	NeuralSpace	0.5603	1 / 24
		IRLab@IITBHU (1)	0.4199	-
		IRLab@IITBHU (2)	**0.5127**	7 / 24
Marathi	1-A	WLV-RIT	0.9144	1 / 25
		IRLab@IITBHU (1)	**0.8545**	12 / 25
		IRLab@IITBHU (2)	0.8410	-
English-Hindi Code-Mix	2	MIDAS-IIITD	0.7253	1 / 16
		IRLab@IITBHU (1)	**0.6795**	6 / 16

hashtag substitution, Maxi-mum sequence length of 128 tokens, and batch size of 32. (Macro F1: 0.7581)

3. **English Sub Task A (submission 3):** mBERT, 25 epochs using emoji and hashtag substitution, replac-ing commonly occurring short-forms like (it's->it is, don't->do not, hahaha->ha etc.),Maximum sequence length of 128 tokens, and batch size of 32. (Macro F1: 0.7812)

4. **English Sub Task A (submission 4):** BERT Large Cased, 25 epochs using emoji, hashtags substitution,Maximum sequence length of 128 tokens, and batch size of 32.(Macro F1: 0.7886)

5. **English Sub Task A (submission 5):** BERT Large Cased, 25 epochs using emoji and hashtags substitu-tion, Maximum sequence length of 128 tokens, and batch size of 16 (Macro F1: 0.7976)

6. **English Sub Task B (submission 1):** BERT Large Cased, 20 epochs using emoji and hashtags substitu-tion, Maximum sequence length of 128 tokens, and batch size of 16 (Macro F1: 0.6093)

7. **Hindi Sub Task A (submission 1):** mBERT, 25 without preprocessing, Maximum sequence length of128 tokens, and batch size of 32 (Macro F1: 0.7471)

8. **Hindi Sub Task A (submission 2):** mBERT, 25 epochs, Maximum sequence length of 256 tokens,using hashtag substitution, and batch size of 16 (Macro F1: 0.7440)

9. **Hindi Sub Task A (submission 3):** mBERT, 25 epochs, using emoji and hashtag substitution, Maxi-mum sequence length of 128 tokens, and batch size of 32. (Macro F1: 0.7547)

10. **Hindi Sub Task B (submission 1):** mBERT, 25 epochs and without using emoji and hashtag substitu-tion, Maximum sequence length of 256 tokens, and batch size of 32 (Macro F1: 0.4199)

11. **Hindi Sub Task B (submission 2):** mBERT, 25 epochs and using emoji and hashtag substitution,Maximum sequence length of 256 tokens, and batch size of 16 (Macro F1: 0.5127)

12. **Marathi Sub Task A (Submission 1):** mBERT, 15 epochs, without using preprocessing of emoji and hashtags substitution, Maximum sequence length of 128 tokens, and batch size of 32(Macro F1: 0.8410)

13. **Marathi Sub Task A (Submission 2):** mBERT, 15 epochs, using preprocessing of emoji and hashtags substitution, Maximum sequence length of 128 tokens, and batch size of 32 (Macro F1:0.8545)

14. **Code Mixed (Submission 1):** Mbert, 15 epochs, using preprocessing of emoji and hashtags substi-tution, Maximum sequence length of 256 tokens, and batch size of 16 (Macro F1: 0.6)

5.7 Discussion

The best-performing binary classification model for English subtask-1A was bert-large-cased using preprocessed data (Submission 5). For the system restrictions, we used 128 tokens as the maximum sequence length for a few phrases with tokenized lengths more than 128. As a result, we had to shorten it, which might explain part of the poor performance. Bert-base-multilingual-cased using preprocessed data was the best-performing model for Hindi subtask-1A. We had to trim the sequence length to 256 as well. Despite the model's comparative score, certain NOT are still incorrectly labelled as HOF. The most likely explanation is because normalising Hashtags, such as #ResignModi to Resign Modi, is considered an assault on a person. It's likely that the presence of any swear words or hate words causes the algorithm to anticipate that the speech will be of type hate. The overall meaning of the line might still be non-hatred, which is difficult to detect and necessitates the discovery of the full context. Furthermore, preprocessing for Marathi does not operate as intended. It also misclassified several people who aren't in the Hall of Fame. The model could not predict PRFN class, as shown in figure 5.4(d) for the multiclass classification on Hindi language submission 1. The table below depicts some of the instances in which our best model recognised incorrect predictions. The gold standard dataset's expected sentiment is compared to the ones predicted by our system in the table's

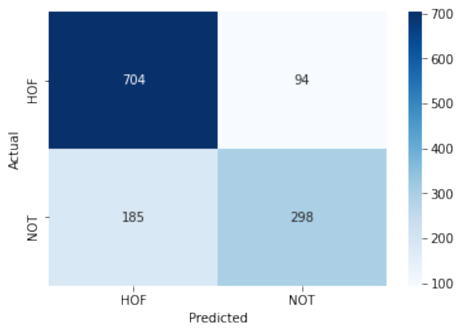
Gold column. It appears that our assessment of public opinion was right.

Table 5.4: Error Analysis

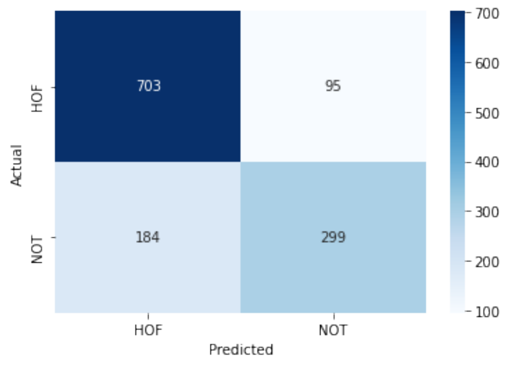
Sample Tweets from dataset	**Gold**	**Predicted**
Saw her shag rug and said ““I can wear that””	HOF	NOT
@bosco_rosco Mate I’m the life and soul of them because I’m not a twat.	NOT	HOF
Just had a phone call from the NHS National immunisation recall centre wanting to discuss my ““ #CovidVaccine plans”” - what the heck are they doing with my phone number ???	HOF	NOT

5.8 Conclusion

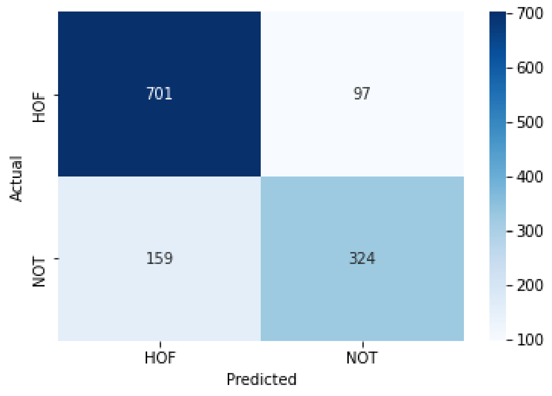
In this paper, we show the method that the IRLab@IITBHU team submitted to the FIRE 2021 shared task HASOC 2021 - Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages. To categorise social media postings in three unique languages and an English-Hindi code mixed language for hate speech, offensive, and objectionable material, our system uses fine-tuning monolingual and multilingual transformer networks. The greatest results are achieved with state-of-the-art transformer models, as demonstrated in the overview paper of the HASOC track at FIRE 2020. Traditional machine learning models are outperformed by pre-trained bi-directional encoder representations employing transformers (BERT). As a result, we’ve solely employed the BERT model with minor pre-processing. We treat each remark as a separate tweet in Subtask 2: Identification of Conversational Hate-Speech in Code-Mixed Languages (ICHCL). We’d like to accomplish this subtask with a graph in the future.



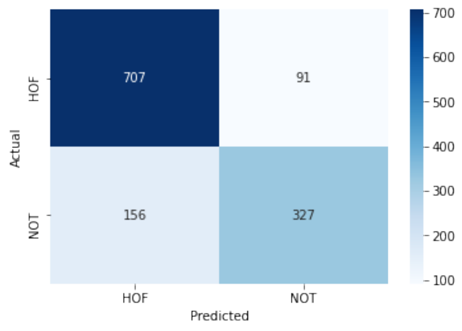
((a)) Submission 1 for Subtask 1A



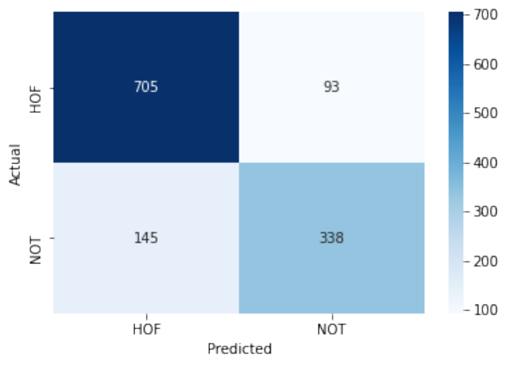
((b)) Submission 2 for Subtask 1A



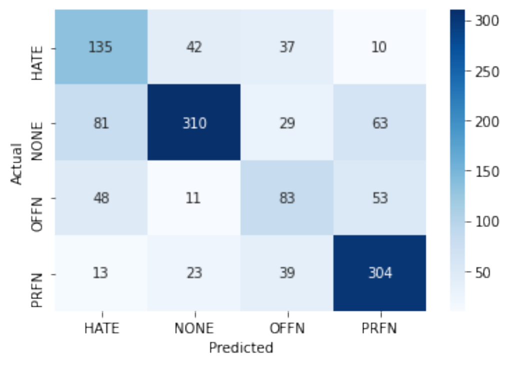
((c)) Submission 3 for Subtask 1A



((d)) Submission 4 for Subtask 1A

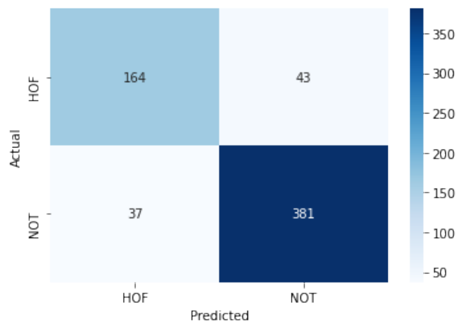


((e)) Submission 5 for Subtask 1A

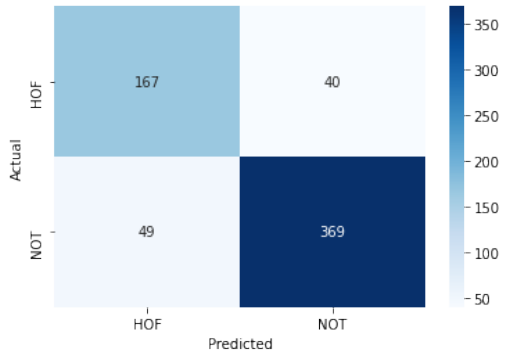


((f)) Submission 1 for Subtask 1B

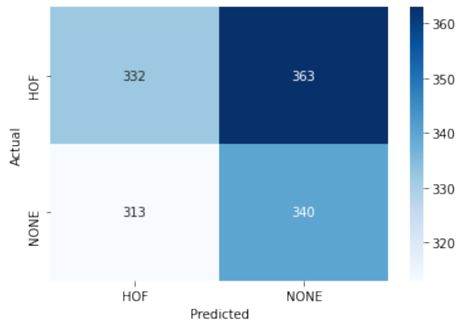
Figure 5.2: Confusion matrix on the given test data for the English language



((a)) Submission 1 for Subtask 1A

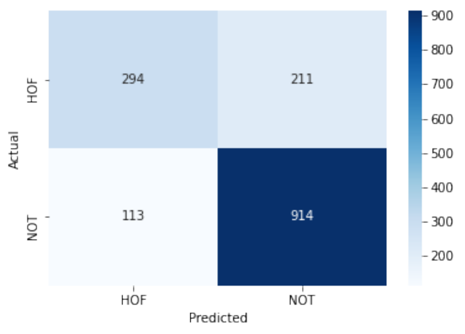


((b)) Submission 2 for Subtask 1A

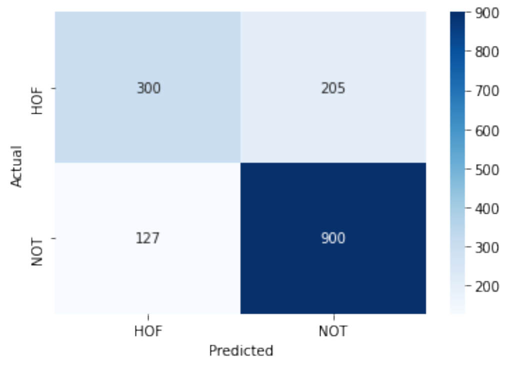


((c)) Submission 1 for Subtask 2

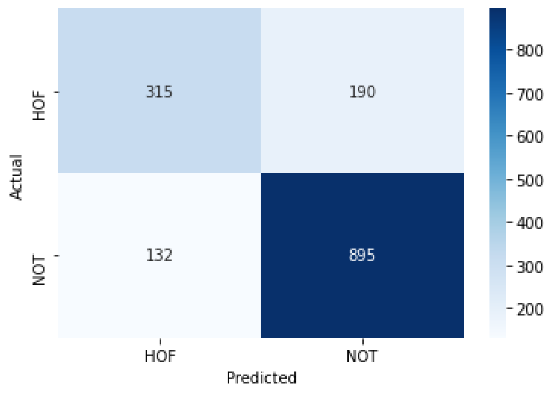
Figure 5.3: Confusion matrix on the given test data for the Marathi language and CodeMix English-Hindi language



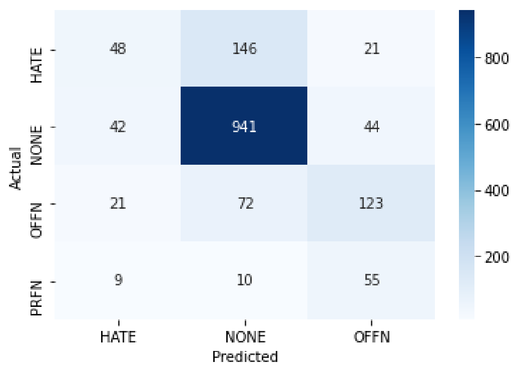
((a)) Submission 1 for Subtask 1A



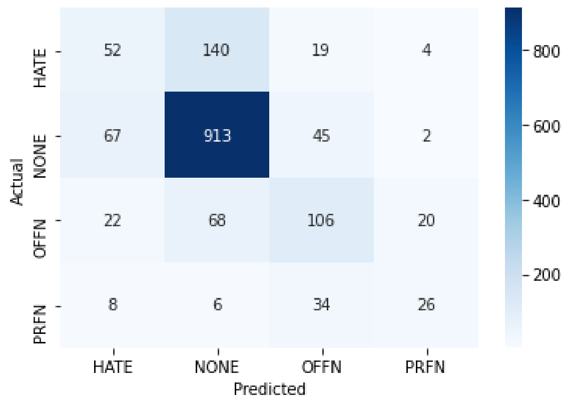
((b)) Submission 2 for Subtask 1A



((c)) Submission 3 for Subtask 1A



((d)) Submission 1 for Subtask 1B



((e)) Submission 2 for Subtask 1B

Figure 5.4: Confusion matrix on the given test data for the Hindi language

Chapter 6

Named Entity Recognition using Meta-Embeddings

6.1 Overview

Switching between the two languages in discussions is more common for things like organisations or products in code-mixing scenarios, which drives us to concentrate on the special topic of Named Entity Recognition (NER) in code-switching scenarios.A Named Entity(NE) is a proper noun, serving as a name for something or someone. Named Entity Recognition (NER) aims to recognize mentions of words which refers to semantic types such as person, location, organization etc. NER is used in a range of natural language processing (NLP) applications, including text interpretation and information retrieval and information extraction (IE).

6.2 Related works

Previous research has primarily focused on applying pre-trained word embeddings from each language with character-level representations to represent noisy mixed-language texts. Despite their success, such word-level techniques overlook the significance of subword-level properties that are shared across languages to assist with this effort, we can use subword-level embeddings like FastText [35]. This will obviously allow us to take advantage of the morphological structure that is shared across languages.We used a multilingual meta-embedding model learned from different languages. Our method can be viewed of as a mechanism for obtaining information from monolingual sources to generate a universal multilingual meta-embedding that can be taught in a supervised manner with code-switching scenarios. Moreover, this is a language-agnostic strategy in which no linguistic information for each word is required. We show the possibility of transferring information from multiple languages to unseen languages. Winata et al. [36] suggested a method for combining multiple sources of data to produce embeddings from several languages. We follow the footsteps of works by Winata, Genta Indra and Lin, Zhaojiang and Fung, Pascale on Learning Multilingual Meta-Embeddings for Code-Switching Named Entity Recognition.We implemented there method on English-Hindi Dataset.

6.3 Dataset

We used English-Spanish Twitter Dataset in CoNLL format and Hindi-English Code-Mixed Text (taken from tweets) for the task of Named Entity Recognition [1].We trained our model using English-Spanish code-switching tweets data from [36]. It has nine entity labels. The labels use IOB format, where every token is labeled as a B-label in the beginning and then an I-label if it is a named entity, or O otherwise.Twitter data is pre-processed and annotated and has 6 NER tags and a 7th O(Other) tag.

Below is the description of both datas in training,development and testing.

[1]https://github.com/SilentFlame/Named-Entity-Recognition/tree/master/Twitterdata

Spanish-English Twitter Data			
	Train	Dev	Test
Total Sentences	50757	832	15634
Total tokens	616069	9583	183011

Table 6.1: Spanish-English Twitter Data

Hindi-English Twitter Data			
	Train	Dev	Test
Total Sentences	1785	551	897
Total tokens	41013	9450	17611

Table 6.2: Hindi-English Twitter Data

6.4 Multilingual Meta Embeddings

To generate multilingual Meta-Embeddings we combine word and subword representations to create a mixture of embeddings. We develop multilingual word and subword meta-embeddings, then concatenate them with character-level embeddings to create final word representations, as illustrated below

Let w denote an n-element word sequence, where $w = [w_1, \ldots, w_n]$. Each word can be tokenized into a list of subwords $s = [s_1, \ldots, s_n]$ and a list of characters $c = [c_1, ..., c_p]$. A function f is used to generate the list of subwords s; $s = f(w)$. The f function converts a word into a list of subwords. Let $E(w)$, $E(s)$, and $E(c)$ be a combination of word, subword, and character embedding lookup tables. There are a variety of monolingual embeddings in each set. Each element in Rd is transformed into an embedding vector. Superscripts (w;s;c) stand for word, subword, and character, and subscripts $f_{i,j}$ stand for element and embedding language index.

6.4.1 Sequence labeling

To predict the entities, we use Transformer-CRF, a transformer-based encoder followed by a Conditional Random Field (CRF) layer. The CRF layer is useful to

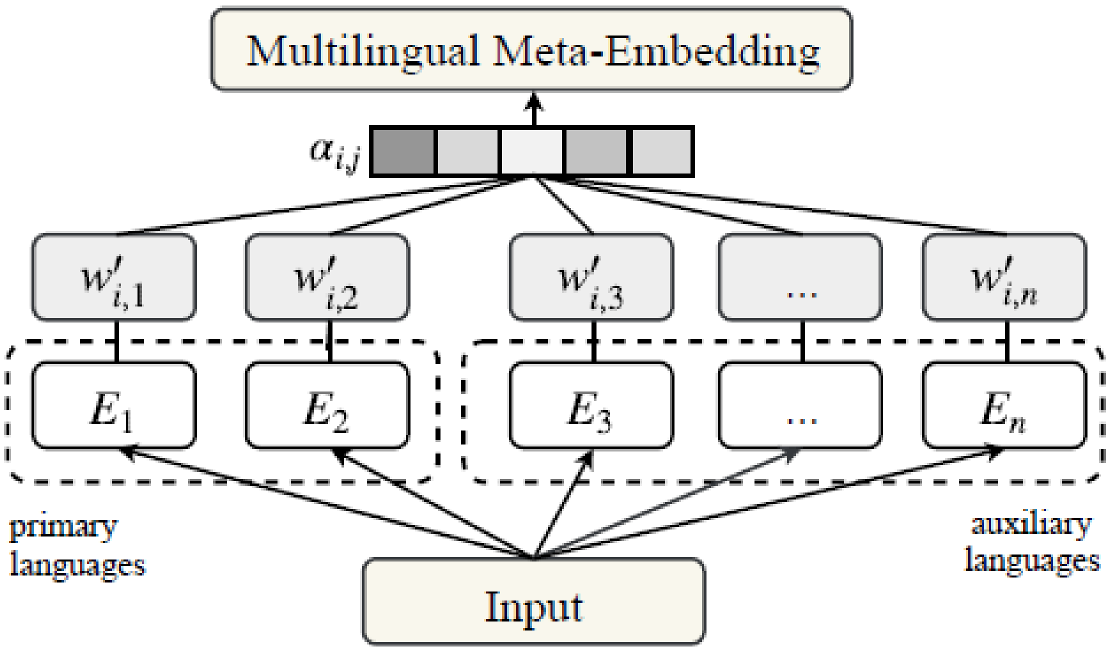


Figure 6.1: Multilingual Meta-Embeddings: The inputs are word embeddings and the output is a single word representation.

constraint the dependencies between labels.

Encoder Architecture

As a sentence encoder, we use a multi-layer transformer encoder:

here W_t is the projection matrix, W_p is the procedural encoding matrix, W_o is the

$$
\begin{aligned}
h_0 &= \text{Concat}(\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_m)\mathbf{W}_t + \mathbf{W}_p, \\
h_l &= \text{Transformer\_blocks}(h_0), \\
o &= h_l \mathbf{W}_o + b_o,
\end{aligned}
$$

Figure 6.2: Equation of our encoder

output layer, h_o is the first layer hidden states, and h_l is the output representation from the final transformer layer. The final layer's output is logits o.

Conditional Random Field

The dependencies between tag labels are calculated using this model. NER necessitates a more stringent limitation, with I-PER following only after B-PER. We use CRF to figure out how the current label and its neighbours are related.In POS tagging, for example, an adjective is more likely to be followed by a noun than a verb, and I-ORG cannot follow I-PER in NER with conventional BIO2 annotation . Instead of decoding each label separately, we use a conditional random field (CRF) to model label sequence jointly. Because we're focusing on sentences rather than individual positions, it's widely assumed that CRF can generate higher tagging accuracy

6.5 Implementation

FastText English (EN) and Spanish (ES) word embeddings were applied, which had previously been trained using CBOW with position-weights in dimension 300, character n-grams of length 5, a window of size 5, and 10 negatives. GloVe Twitter English embedding is also included, as it is one of the most extensively used count-based embedding models.Subword Embedding of EN and ES are also used in case of Multilingual Meta-embedding. To learn the embeddings, it performs matrix factorization on the co-occurrence matrix of words. Our Meta-embedding was created using primary and auxiliary languages.For English-Hindi Data Fasttext Hindi(HI) is used with English(En) Fasttext Embeddings.

We used following strategies to use pre trained embeddings -

6.5.1 CONCAT

We concatenate multiple word embeddings by merging the dimensions of word representations of different embeddings. This method combines embeddings into a high dimensional input.

$$\mathbf{x}_i^{CONCAT} = [\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,n}].$$

Figure 6.3: CONCAT equation

6.5.2 LINEAR

We sum all word embeddings into a single word vector with equal weight. This method combines embeddings without considering the importance of each of them.This method is efficient since it does not increase the dimensionality of the input. We apply a projection layer through $w_{i,j}$ to have equal dimension before we sum.

$$\mathbf{w}_i^{LINEAR} = \sum_{j=0}^{n} \mathbf{w'}_{i,j},$$
$$\mathbf{w'}_{i,j} = \mathbf{a}_j \cdot \mathbf{w}_{i,j} + b_j,$$

Figure 6.4: LINEAR equation

6.5.3 Multilingual Meta-Embedding

We generate meta-representations by combining the vector representations of numerous monolingual pre-trained embeddings in various subunits like word and subword. The dimensions of the original space are transformed into a new shared space using a projection matrix W_j.

6.6 Experiment and Results

Experiment details

Our model contains four layers of transformer encoders with a hidden size of 200, four heads, and a dropout of 0.1. Four layers of transformer encoders with a concealed

$$\mathbf{x'}_{i,j} = \mathbf{W}_j \cdot \mathbf{x}_{i,j}, \quad (1)$$

$$\alpha_{i,j} = \frac{\exp(\phi(\mathbf{x'}_{i,j}))}{\sum_{k=1}^{n} \exp(\phi(\mathbf{x'}_{i,k}))}, \quad (2)$$

$$\mathbf{u}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{x'}_{i,j}. \quad (3)$$

Figure 6.5: Multilingual Metaembedding

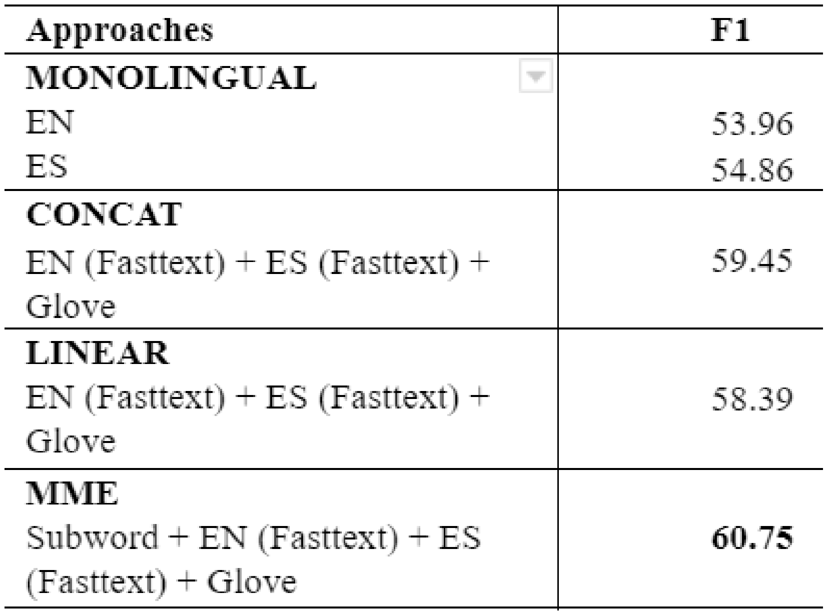
size of 200, four heads, and a dropout of 0.1 make up our model. We utilise the Adam optimizer and begin the training with a learning rate of 0.1 and a 15-iteration early stop. User hashtags and mentions are replaced with USR, emoji with EMOJI, and URL with URL.

In case of English -Spanish data we validated our model on the training and development sets since we lacked test labels. As our submission for each subtask, we chose the top models from each evaluation.The absolute F_1 score is used to evaluate our model.

For Hindi-English Data we had test labels so we evaluate our model using Evaluation metrics like precision,recall and accuracy.

Hindi-English Twitter Data			
	CONCAT	LINEAR	MME
F1 Score	0.6654	0.6528	0.6890

Table 6.3: F1 scores for Named Entity Tags for ENG-HIN Data where CONCAT is (EN(Fasttext)+Hindi(Fasttext)), LINEAR is EN(Fasttext)+Hindi(Fasttext)), MME is (Subword(EN+HI)+EN(Fasttext)+Hindi(Fasttext))



Approaches	F1
MONOLINGUAL	
EN	53.96
ES	54.86
CONCAT	
EN (Fasttext) + ES (Fasttext) + Glove	59.45
LINEAR	
EN (Fasttext) + ES (Fasttext) + Glove	58.39
MME	
Subword + EN (Fasttext) + ES (Fasttext) + Glove	**60.75**

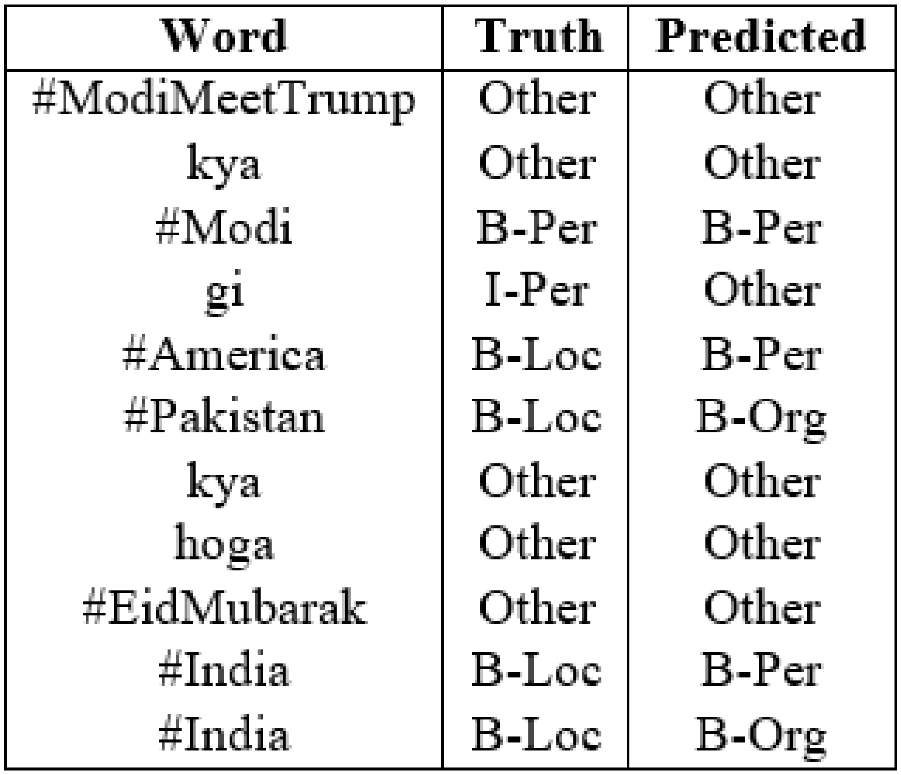
Figure 6.6: F1 scores of tokens tagged as Named Entity,here EN is English and ES is Spanish

Hindi-English Twitter Data			
	CONCAT	LINEAR	MME
Accuracy	92%	91%	93%

Table 6.4: Accuracy for all tags of ENG-HIN Data where CONCAT is (EN(Fasttext)+Hindi(Fasttext)), LINEAR is EN(Fasttext)+Hindi(Fasttext)), MME is (Subword(EN+HI)+EN(Fasttext)+Hindi(Fasttext))

6.6.1 Discussion

Our approach performs well with the ’Loc’ and ’Other’ tags, providing accurate predictions. Because our corpus contains names like ’Location,’ ’Organization,’ and ’Person,’ it’s challenging for our machine learning models to learn the right tags for these names. For example, in our annotation, ’Hindustan’ is labelled as ’B-Loc’ and ’Hindustani’ is labelled as ’B-Org’ because the former is one of the five names for the country India while the latter represents the citizens, making it a group representation that we used for Organization. As a result, lexically similar words with different tags make our model’s learning phase challenging, resulting in some inaccurate token tagging.



Word	Truth	Predicted
#ModiMeetTrump	Other	Other
kya	Other	Other
#Modi	B-Per	B-Per
gi	I-Per	Other
#America	B-Loc	B-Per
#Pakistan	B-Loc	B-Org
kya	Other	Other
hoga	Other	Other
#EidMubarak	Other	Other
#India	B-Loc	B-Per
#India	B-Loc	B-Org

Figure 6.7: An Example Prediction of our Model

6.7 Conclusion

We used Multilingual Meta-Embeddings (MME) for learning how to merge several monolingual word-level and subword-level embeddings. Using monolingual pretrained embeddings to create multilingual representations. MME eliminates language identification dependencies in code-switching Named Entity Recognition tasks by incorporating additional information from semantically related embeddings.On the problem of Named Entity Recognition for English-Spanish code-switching data, we reach state-of-the-art results. We also show that our model can efficiently use subword information from languages of various origins to build better word representations.

Chapter 7

Publications communicated from this thesis work

- Supriya Chanda, S Ujjwal, Shayak Das and Sukomal Pal. **Fine-tuning Pre-Trained Transformer based modelfor Hate Speech and Offensive Content Identificationin English, Indo-Aryan and Code-Mixed(English-Hindi) languages**, *FIRE'21: Forum for Information Retrieval Evaluation, Dec 13–17, 2021*
- Supriya Chanda, Sukomal Pal, S Ujjwal. **Transformer Based Language Identification from social media code-mixed data**, *Journal of Intelligent Information Systems.*

Bibliography

[1] D. K. Prabhakar, S. Pal, and C. Kumar, "Query expansion for tansliterated text retrieval," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 1, no. 1, 2021. [Online]. Available: https://doi.org/10.1145/3447649

[2] R. Joshi and R. Joshi, "Evaluating input representation for language identification in hindi-english code mixed text," *ArXiv*, vol. abs/2011.11263, 2020.

[3] B. Barnett, E. Codo, E. Eppler, M. Forcadell, P. Gardner-Chloros, R. van Hout, M. Moyer, M. Torras, M. Turell, M. Sebba, M. Starren, and S. Wensink, "The lides coding manual - a document for preparing and analyzing language interaction data. version 1.1, july 1999," *International Journal of Bilingualism*, vol. 4, pp. 131–270, 2000, pagination: 140.

[4] G. A. Guzman, J. Serigos, B. E. Bullock, and A. J. Toribio, "Simple tools for exploring variation in code-switching for linguists," in *Proceedings of the Second Workshop on Computational Approaches to Code Switching.* Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 12–20. [Online]. Available: https://www.aclweb.org/anthology/W16-5802

[5] B. Gambäck, "On measuring the complexity of code-mixing," 2014.

[6] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in Pre-Training Distributed Word Representations," in *Proceedings of the In-*

ternational Conference on Language Resources and Evaluation (LREC 2018), 2018.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *Proceedings of the 2019 Conference of the North*, 2019.

[8] J. Allan, B. Croft, A. Moffat, and M. Sanderson, “Frontiers, challenges, and opportunities for information retrieval: Report from swirl 2012 the second strategic workshop on information retrieval in lorne,” *SIGIR Forum*, vol. 46, no. 1, p. 2–32, may 2012. [Online]. Available: https://doi.org/10.1145/2215676.2215678

[9] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei, “Rumor has it: Identifying misinformation in microblogs,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP ’11. USA: Association for Computational Linguistics, 2011, p. 1589–1599.

[10] S. Karimi, F. Scholer, and A. Turpin, “Machine transliteration survey,” *ACM Comput. Surv.*, vol. 43, no. 3, apr 2011. [Online]. Available: https://doi.org/10.1145/1922649.1922654

[11] J. Zobel and P. Dart, “Phonetic string matching: Lessons from information retrieval,” in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’96. New York, NY, USA: Association for Computing Machinery, 1996, p. 166–172. [Online]. Available: https://doi.org/10.1145/243199.243258

[12] S. MacAvaney, L. Soldaini, and N. Goharian, “Teaching a new dog old tricks: Resurrecting multilingual retrieval using zero-shot learning,” *Advances in Information Retrieval*, vol. 12036, p. 246, 2020.

[13] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, "End-to-end neural ad-hoc ranking with kernel pooling," *CoRR*, vol. abs/1706.06613, 2017. [Online]. Available: http://arxiv.org/abs/1706.06613

[14] R. Pradeep, R. Nogueira, and J. Lin, "The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models," *CoRR*, vol. abs/2101.05667, 2021. [Online]. Available: https://arxiv.org/abs/2101.05667

[15] Z. Dai and J. Callan, "Context-aware sentence/passage term importance estimation for first stage retrieval," *CoRR*, vol. abs/1910.10687, 2019. [Online]. Available: http://arxiv.org/abs/1910.10687

[16] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson, "Terrier information retrieval platform," vol. 3408, 01 2005, pp. 517–519.

[17] C. Macdonald and N. Tonellotto, "Declarative experimentation ininformation retrieval using pyterrier," in *Proceedings of ICTIR 2020*, 2020.

[18] J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore, "Learning from bullying traces in social media," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, ser. NAACL HLT '12. USA: Association for Computational Linguistics, 2012, p. 656–666.

[19] P. Burnap and M. L. Williams, "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy & Internet*, vol. 7, no. 2, pp. 223–242, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/poi3.85

[20] M. Wiegand, M. Siegel, and J. Ruppenhofer, "Overview of the germeval 2018 shared task on the identification of offensive language," ser. Proceedings of

GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018), Vienna, Austria – September 21, 2018. Vienna, Austria: Austrian Academy of Sciences, 2018, pp. 1 – 10. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:bsz:mh39-84935

[21] T. Davidson, D. Warmsley, M. W. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” *CoRR*, vol. abs/1703.04009, 2017. [Online]. Available: http://arxiv.org/abs/1703.04009

[22] R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri, “Benchmarking aggression identification in social media,” in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018).* Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1–11. [Online]. Available: https://www.aclweb.org/anthology/W18-4401

[23] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, “Hate speech detection with comment embeddings,” in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW ’15 Companion. New York, NY, USA: Association for Computing Machinery, 2015, p. 29–30. [Online]. Available: https://doi.org/10.1145/2740908.2742760

[24] I. Kwok and Y. Wang, “Locate the hate: Detecting tweets against blacks,” in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, ser. AAAI’13. AAAI Press, 2013, p. 1621–1622.

[25] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, “Abusive language detection in online user content,” in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW ’16. Republic and Canton of Geneva, CHE: International World Wide Web

Conferences Steering Committee, 2016, p. 145–153. [Online]. Available: https://doi.org/10.1145/2872427.2883062

[26] K. Dinakar, R. Reichart, and H. Lieberman, “Modeling the detection of textual cyberbullying,” in *The Social Mobile Web, Papers from the 2011 ICWSM Workshop, Barcelona, Catalonia, Spain, July 21, 2011*, ser. AAAI Workshops, vol. WS-11-02. AAAI, 2011. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/3841

[27] A. Saroj, S. Chanda, and S. Pal, “IRlab@IITV at SemEval-2020 task 12: Multilingual offensive language identification in social media using SVM,” in *Proceedings of the Fourteenth Workshop on Semantic Evaluation.* Barcelona (online): International Committee for Computational Linguistics, Dec. 2020, pp. 2012–2016. [Online]. Available: https://aclanthology.org/2020.semeval-1.265

[28] H. Mubarak, K. Darwish, and W. Magdy, “Abusive language detection on Arabic social media,” in *Proceedings of the First Workshop on Abusive Language Online.* Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 52–56. [Online]. Available: https://www.aclweb.org/anthology/W17-3008

[29] H.-P. Su, Z.-J. Huang, H.-T. Chang, and C.-J. Lin, “Rephrasing profanity in Chinese text,” in *Proceedings of the First Workshop on Abusive Language Online.* Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 18–24. [Online]. Available: https://www.aclweb.org/anthology/W17-3003

[30] B. Ross, M. Rist, G. Carbonell, B. Cabrera, N. Kurowsky, and M. Wojatzki, “Measuring the reliability of hate speech annotations: The case of the european refugee crisis,” *CoRR*, vol. abs/1701.08118, 2017. [Online]. Available: http://arxiv.org/abs/1701.08118

[31] D. Fišer, T. Erjavec, and N. Ljubešić, “Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in Slovene,” in *Proceedings of the First Workshop on Abusive Language Online.* Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 46–51. [Online]. Available: https://www.aclweb.org/anthology/W17-3007

[32] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, “Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval),” in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 75–86.

[33] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, and c. Çöltekin, “SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020),” in *Proceedings of SemEval*, 2020.

[34] M. Dadvar, D. Trieschnigg, R. Ordelman, and F. de Jong, “Improving cyberbullying detection with user context,” in *Advances in Information Retrieval*, P. Serdyukov, P. Braslavski, S. O. Kuznetsov, J. Kamps, S. Rüger, E. Agichtein, I. Segalovich, and E. Yilmaz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 693–696.

[35] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, “Learning word vectors for 157 languages,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://aclanthology.org/L18-1550

[36] G. I. Winata, Z. Lin, and P. Fung, “Learning multilingual meta-embeddings for code-switching named entity recognition,” in *Proceedings of the 4th*

Workshop on Representation Learning for NLP (RepL4NLP-2019). Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 181–186. [Online]. Available: https://www.aclweb.org/anthology/W19-4320