IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 16, 2021, accepted July 29, 2021, date of publication August 11, 2021, date of current version September 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3104106

Transformer Based Language Identification for Malayalam-English Code-Mixed Text

S. THARA AND PRABAHARAN POORNACHANDRAN
Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, Kerala 690525, India
Corresponding author: S. Thara (thara@am.amrita.edu)

ABSTRACT Social media users have the proclivity to write majority of the data for under resourced languages in code-mixed format. Code-mixing is defined as mixing of two or more languages in a single sentence. Research in code-mixed text helps apprehend security threats, prevalent on social media platforms. In such instances, language identification is an imperative task of code-mixed text. The focus of this paper is to carry out a word-level language identification (WLLI) of Malayalam-English code-mixed data, from social media platforms like YouTube. This study was centered around BERT, a transformer model, along with its variants - CamemBERT, DistilBERT - for intuitive perception of the language at the word-level. The propounded approach entails tagging Malayalam-English code-mixed data set with six labels: Malayalam (mal), English (eng), acronyms (acr), universal (univ), mixed (mix) and undefined (undef). Newly developed corpus of Malayalam-English was deployed for appraisal of the effectiveness of state-of-the-art models like BERT. Evaluation of the proffered approach, accomplished with other code-mixed language such as Hindi-English, notched a 9% increase in the F1-score.

INDEX TERMS Natural language processing, language identification, bidirectional encoder representations from transformers (BERT), text mining, corpus preparation, deep learning, code-mixed.

I. INTRODUCTION

The advent of the current millennium has been accompanied by a phenomenal increase in usage of Indian language in multilingual interpersonal communications such as news forums, webpages, email, and social-media chats. The widespread proliferation of mobile phones has sparked seemingly relentless usage of popular micro-blogging sites - Linkedin, Facebook, Twitter, and WhatsApp - to convey opinions, comments and impressions in natural language processing (NLP) [1]. People routinely express their interests and viewpoints on product reviews, movies, hotels, and commodities. Enabled by computer-mediated communications, multilingual speakers often switch between languages, amidst the spoken or written world of information exchange. Devoid of rigid structure and functional expectations, research in interactive conversational data poses an arduous challenging task.

India has an enriched heritage of linguistics, influenced by their close bonding with the English language, thereby enabling India to become the second largest population of English language speakers. Malayalam is one of the Dravidian languages spoken in the southern region of India, with official designation in the Indian state of Kerala and union territories of Lakshadweep and Puducherry. There are nearly 38 M Malayalam speakers across the globe. It is a highly agglutinative language. Vatteluttu, the Malayalam script, is extended with symbols from the Grantha script. Vatteluttu, a member of the Abugida writing system, is alpha-syllabic, which is partly alphabetic and partly syllable-based.

TABLE 1. An example of code-mixed text.

Malayalam-English code-mixed text										
Text	entha	full	glass	vechond	irkunath	enthlum	problem	undo.		
Tags	mal	eng	eng	mal	mal	mal	eng	mal.		

In Table 1, the Malayalam words tagged as mal were written in the Roman script instead of the native Malayalam script. Code-mixed text [2] is often quite vague, imprecise and unpredictable; in the above example, a non-Malayali person reading the word 'undo' gets puzzled about the source

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

of its language - English, unknown lingo, slang, jargon or Malayalam! In such scenarios, language identification is a primary imperative of code-mixed text [3]. Comments or tweets obtained from social media sites can be rendered noisy or unreadable, by resort to nonstandard abbreviations, misspelled words, typos, undefined symbols, emoticons, hashtags – all of which complicate the normalization process.

The key problem of interest for which this study was undertaken is that a sizable majority of the data available in social media, for the under-resourced languages [4] is code-mixed (mixing of two or more languages in a single sentence, or even a single word). Social media enthusiasts often adopt Roman script for typing it because of the ease of its input in this manner. Hence, deciphering the language is an inevitable prerequisite task for any code-mixed text [5], apropos of NLP [6]. Therefore, code-mixing can be considered as an engineering problem, with optimization of existing algorithms, but with a little innovation of new methods leads to a higher level of performance.

The principal schemes for WLLI started off with computational linguistics [7], [8] where sociological and conversational necessities in code-mixed corpus are discussed. It continued with marshalling machine learning based [9]–[11] approaches, using N-gram with weights [12] and dictionary based [13] feature extraction methods, for language identification. Few other works have leaned on deep learning [14]–[16] and hybrid based approaches [17]. Various papers [18]–[20] have called for the creation of a code-mixed corpus, due to unavailability of a publicly accessible gold standard data set, for research purposes. Recent works [21], [22] have delineated the impediments entailed in the preparation of code-mixed corpus for Indian languages.

Hitherto, bidirectional encoder representations from transformers (BERT) and none of the state-of-the-art (SOTA) models [23] adopted WLLI with Malayalam-English code-mixed data. In fact, this paper proposes a futuristic word level collection of language identifiers for Malayalam-English code-mixed corpus, with requisite labels. Besides, the paper delineates a fusion of BERT variants for WLLI of code-mixed Indian language.

The challenges of tagging the dataset.

1) A sense of variability exists to decide the language tag of a particular token when it shares similar words in both languages such as English and Malayalam. Confusion arises for tagging certain words, such as *undo*, which if perceived merely as a standalone word, is tagged as *eng*, whereas considering its meaning in context, it should be tagged as *mal*.
2) The pre-processing stage includes suppression of extraneous successive characters in words, which occur more than twice in a single word, such as '*cooool*' [instead of cool], '*greaaaat*' [instead of great].
3) There are situations where words [whole or partial words, including prefix and suffixes] from two languages are used within a single word. It is very difficult to observe a clear difference between the mixed languages in a single word. For e.g.: '*groupil*', '*leftil*' are considered as mixed with English and Malayalam vocabulary.
4) Contracted word forms, from user-generated content of social media, are quite often ambiguous and can represent different standard word forms, such as '*bght*' may refer to brought or bought.
5) We have to deal with unbalanced data distribution, since a majority of words from social media sites may be labelled as English and Malayalam and very few words labelled as universal and acronym.

Key contributions of the paper are noted hereunder:

- Innovative approach for development of a 50K code-mixed Malayalam- English, code-mixed corpus, tagged with six labels: Malayalam (mal), English (eng), undefined (undef), mixed (mix), acronyms (acr), universal (univ).
- Applying SOTA model like BERT on the newly developed corpus for WLLI task.
- Bidirectional Encoder Representations from Transformer (BERT) - based models and their variants - CamemBERT, DistilBERT, ELECTRA, XLM-RoBERTa are used for WLLI.
- The proffered method, when compared with other benchmark code-mixed data set, achieved an overall increase of F1-score by 9%.

Rest of this paper is organized as follows. Section 2 gives an overview of the related works, followed by a definition of the problems addressed in this study. Section 3 describes the proposed approach that includes data generation procedure and the algorithms used for WLLI problems. Section 4 discusses the results and the inferences from this study. Section 5 gives limitations of the study. Finally section 6 concludes the paper with closing remarks.

II. RELATED WORKS

Researchers gave importance to a wide range of machine learning algorithms using the Bayesian approach (BN) [24], maximum entropy classifier (MaxEnt) [25], multinomial Naive Bayes classifier (MNB), conditional random field (CRF) [26], support vector machines (SVM) classifier [27], and decision trees (DT). Some of the recent studies have presented unbiased measures of code-switching from micro-blogging sites, which is helpful for comparison among various types of code-mixed corpora [13], [28], [29].

Several researchers focused on language identification task with basic feature engineering methods and classification techniques. Veena *et al.* [30] word-based and character-based embedding representations, for carrying out WLLI of code-mixed data. Types of embedding were 1-gram, 3-gram and 5-gram with the maximum accuracy of 95% for 5-gram embedding. Sequiera *et al.* [31] developed code-mixed sentences for Hindi-English pair, and performed annotation at fine-grain level, accounting for named entities, POS tagging and normalization. Bansal *et al.* [32] proposed

language identification by the use of three classifiers: Gaussian Naive Bayes, Logistic Regression and Decision Tree. Character n-gram features were extracted and tested with these 3 classifiers, which yielded high accuracy scores.

Researchers also gave significance to linguistic features and dictionary based approach while identifying languages in code-mixed domain. Guzman *et al.* [33] developed a LSTM model for artificial immune systems, inspired by biological immune systems. These LSTM classification models are used for language identification in code-mixed text. The features identified to tag the words are the alignment information which are based on the phonemic mechanism. Recently, Adouane and Dobnik [34] solved a sequence tagging problem, using the standard Hidden Markov Model (HMM) approach, and identified n-grams for feature extraction. A lexicon-based approach has also been experimented. Linguistic rules were formulated to tackle vague and unfamiliar words.

Recent studies [35]–[37] reveal the utility of code-switching in Hindi-English language pairs, from Twitter, Facebook, and WhatsApp social media sites. King *et al.* [38] proposed the generation of a candidate sentence as feature generation, for the Bidirectional Long Short Term Memory (Bi-LSTM) - based neural network, to classify Hindi-English code-mixed texts into three labels: positive, negative and neutral. The classification step was preceded by preprocessing stages - removal of hyperlinks, emoticons, mentions and hashtags, from the sentences. Jaech *et al.* [39] applied a hierarchical neural model to the task of language identification. This team's proposed system has two components: a conventional neural network to perform char2vec, where a continuous representation vector is built for each word; the second component is a Bi-LSTM, where the embedding vectors are processed to output a vector, for each word in a tweet.

Recent works [40] have proposed diverse paths for WLLI. One of them used a multichannel [four channels] neural network, which included three convolutional 1-D networks and a LSTM network; it was tested on Bengali-English and Hindi-English language pairs. The model achieved impressive accuracy scores of 93.28% on the Bengali data set, and 93.32% on the Hindi data set. Another approach used character encoding and root phone encoding [40] to train LSTM models. Jamatia *et al.* [41] addressed WLLI of code-mixed language pairs - English-Hindi and Bengali-English - from Facebook, Twitter, WhatsApp social media sites. Their paper leaned on a baseline approach - CRF classifier, and deep learning-based approaches like LSTM and Bi-LSTM, where the latter outperformed the former by 2%, since CRF experiments are run on top of it.

The author in [42] presented named entity recognition in Hindi-English tweets, where each tweet was annotated at a word-level, for three named entities - person, location, and organization. The convolution neural network (CNN) and LSTM based proposed models, had scores greater than the baseline Named Entity Recognition (NER) model by an F1-score of 33.18%. Markovič *et al.* [43] discussed the structure and complexity of texts in Slovene belles-lettres, with an emphasis on evaluating the differences in the texts for different age groups.

Transformer based approach also paved way for language identification task. Samih *et al.* [44] developed a BERT architecture for detection of sarcasms from the Reddit and Twitter online platforms. This model gives a better representation of contextual information, outperforms the sequential models. It gave an accuracy of 74% in Twitter data set and 63% in Reddit data set. For identification of offensive language, Ranasinghe *et al.* [45] described neural network architectures such as LSTM, GRU and BERT, a transfer learning-based model. Pursuant to a comprehensive assessment of text classification measures, BERT turned out to be the best performer for the English, Hindi, and German data sets provided to the participants.

Several works are published for other language pairs. Molina *et al.* [17] proposed labeling each word into one of these six labels: lang1 - Modern Standard Arabic (MSA), lang2 - Dialectal Arabic (DA), other, mixed, ambiguous and named entities. The machine learning (ML) method used is CRF and received an F1-score of 89%. In [17], the author described a shared task of language identification, on code-switched data of Spanish-English (SPA-ENG) and Modern Standard Arabic-Dialectal Arabic (MSA-DA) language pairs. The task was to label each word with 8 tags: lang1, lang2, fw, mixed, other, ne, ambiguous, and unk. The results showed F1 scores of 91.3% for the SPA-ENG language pair and 83.0%, for the MSA-DA pair. Patwa *et al.* [46] collected 20K labelled tags for Hindi-English language pair and 19K labelled tags for Spanish-English language pairs, for the sentiment analysis of code-mixed dataset, resulting in F1-score of 75% and 80.6% for Hindi-English and Spanish-English dataset respectively.

In paper [47] described a framework for online question answering system, whose components include input, the language translation, the answer type, the web search, the NER part and the results section. Researchers have also exploited other areas of code-switching schemes, like information retrieval systems e.g.: Automatic Aspect Extraction [48], Polarity Identification [49], Sequence labelling tasks such as Named Entity Recognition [50], Automatic Speech Recognition [51], and Parsing [52]. From the related works we observed that many studies have been conducted in the language identification task in many language pairs but none of them are addressing Malayalam-English due to lack of publicly available data set. The proposed work solves the problem of WLLI in the Malayalam-English code-mixed using transformer based approach. Hence this paper gives the motivation to start research in this direction.

III. PROPOSED APPROACH

Fig. 1 shows the flow diagram of the proposed strategy for WLLI. The WLLI constituent stages are data acquisition, data preprocessing, data annotation and language identification.

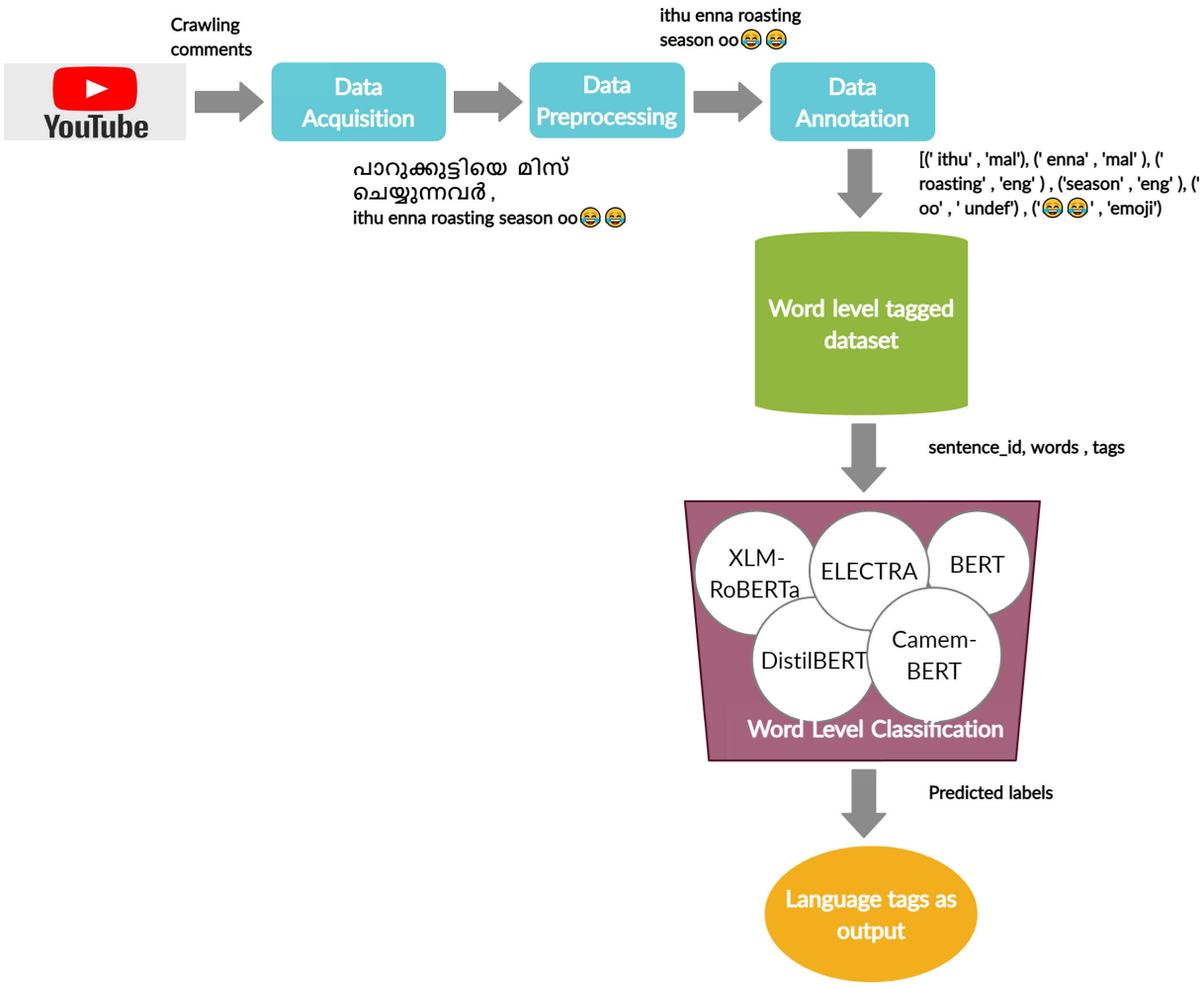

FIGURE 1. Flow diagram of the proposed work shows stages such as data collection, data annotation, word level classification and predicted labels. Intermediate output of various stages is also presented.

On any social media platform, the primary concern with texting is the identification of the language(s) used in the message, as users may neither recognize nor understand the bewildering diverse languages, across the globe. Given the paucity of task-specific human-labelled training examples, this study is focused on the development of a WLLI system, for a Malayalam-English code-mixed data set. The proffered approach involves classification of each word into its corresponding language. The description of the six annotated labels are given in Table 2. Around 50 K code-mixed sentences were collected, with an aggregate of 775 430 words.

This study highlights the development of the first-ever collection of word level language identifiers, as no openly accessible benchmark corpus has been reported, for the Malayalam-English code-mixed data set. This initiatory developed data set is composed of 50 K sentences, extracted from YouTube, which include genres like politics, sports, entertainment, news, and food. A systematic scrupulous procedure was undertaken for the collection of Malayalam-English code-mixed data and annotation schemes.

A. DATASET GENERATION PROCEDURE

In this section, a comprehensive approach is introduced for the formulation of Malayalam-English code-mixed corpus. With the help of the YouTube social media platform, an annotated data set was prepared for WLLI of Malayalam-English codemixed data set. In current times, as part of their daily routine, billions of people interact on social media platforms, dispersed across the globe, to post comments, upload status and videos. With due consideration of this likelihood, the codemixed data set was developed in three stages. The first stage involves Data Acquisition, a major step that deals with the authentication of social media platforms, and keeping Gmail accounts in sync. Channel_id is fed as input in the data acquisition stage, to retrieve the comments of

TABLE 2. Corpus statistics for the language pair Mal-Eng in WLLI.

Labels	Description	Tokens
eng	English words only	320 353 (41.31 %)
mal	Malayalam words only	307 858 (39.7 %)
acr	Acronym words only	9785 (1.26 %)
univ	Special characters, emoticons, and numbers	89 385 (11.5 %)
mix	Code-switched morphemes or combination of Malayalam and English words	24 820 (3.2 %)
undef	Undefined or unintelligible words	23 229 (2.99 %)

YouTube channels. The output of the first stage is comprised of Roman scripts, native scripts and bilingual sentences. The next stage is Data Preprocessing, entailed with the extraction of bilingual (Malayalam - English) sentences only. The output of the second stage consists of bilingual sentences of the Malayalam - English language pair. Data Annotation, the last stage, marshals task concepts such as tokenization, POS_tagging, and lemmatization, for tagging code-mixed text. The output of the third and final stage is a tagged word-level code-mixed that contains a list of tuples (word, tag). Detailed explanations are noted in the sub-sections.

1) DATA ACQUISITION

Social media platforms play a key role in the extraction of raw text for the development of annotated corpora. Among all the social media platforms, YouTube was considered, to build this corpus in four primary domains - movies, news, food-reviews and entertainment –for retrieval of code-mixed comments. The various sub modules used for data acquisition process include:

- **Google-api-python client:** Google Application Programming Interface (API) facilitates programmatic access to Google platforms like Google Drive. This API simplifies the coding process. If a user needs access to large data source files for training the data, uploading large files, each access entails an avoidable time-intensive task. This API is useful for upload of the data sets and their access, via a Google Drive link.
- **Google-auth, google-auth-oauthlib and google-pauth-httplib2:** These libraries help users with a secure access to files, for the setup of authorization and authentication. The authorization flow was designed through a command-line argument in Google Colab platform.
- **BeautifulSoup4:** BeautifulSoup4 helps in the scraping of information from webpages. It parses the scraped information into HTML or XML format, which enables easier manipulation of the data that can be interpreted by humans.
- **Emoji:** This module helps user's implementation of emojis by predefined functions, or in the identification of emojis in a data set, via regular expressions. Every emoji has a unique identification code; emojize( ) is the predefined function that prints the emoji by passing name(''thums up:'') through a parameter.
- **Google.oauth2.credentials:** This library provides the credentials for authorization security. Each and every time it refreshes and generates the new tokens (passwords) while we are accessing our files in third-party apps like Colab and Jupyter.
- **Natural Language Tool Kit:** The Natural Language Tool Kit (NLTK) module, a constituent of NLP application, is one of the leading module that works on human readable data. Diverse submodules of NLTK help with pre-processing data sets like Tokenization, removal of stop words, Stemming, Lemmatization, and easier identification of texts by regular expressions.
- **Googleapiclient.errors:** This module helps in the diagnosis of client-side errors, such as uploading a file from a third-party source can trigger errors due to network issues, non-compliant data formats (meta - 64, meta -32), service restrictions; all these errors are handled with HTTP error codes of 400, 404, 200, etc.
- **RegexTokenizer:** Regex plays a key role during preprocessing steps of NLP routines.
- **String:** This module contains a number of functions like string.ascii_lowercase, string.ascii_uppercase, string.digits, and string.hexdigits. Some of these functions may work on regex (Regular Expressions) str.split() splitting the sentences into specified condition.
- **Files:** This module is safe and secure to upload files from a local disk, while working on different platforms like Google's Colab [or Colaboratory]. Files supports diverse file extensions (.json,.txt,.vlc,.xml,.csv) and data formats (meta, hexa, Unified Text Format ((UTF)).
- **Pandas:** Pandas is one of the powerful libraries in Python that helps in the creation of a series of data frames of *rows* × *dimension* ($m \times n$) dimension. Each row having an index will help retrieval of the particular lines of data from a data frame. Predefined functions manipulate the data by insertion, deletion etc.

The following procedure 1 illustrates how data is acquired from a YouTube channel:

Procedure 1: How comments are retrieved from YouTube channel.

1) A Google API key needs to be uploaded, to be connected with social media platforms, in order to retrieve any type of data for manipulation (tagging/ sentiment). The API key is unique to every platform.

We are working with a YouTube platform, the API key for YouTube, ''youtube-control-api-key.json'', is in JavaScript Object Notation (JSON) format. The 428 byte long API key helps to connect the YouTube interface.

2) To retrieve comments from social media platforms like YouTube, the user needs to be authenticated with the YouTube sync G-mail account.
   a) Check, if the authentication process is completed.
   b) If authentication is unsuccessful then repeat Step 1 or terminate/quit the program and there is no limit on the number of permissible attempts.
3) Select any YouTube channel_id and give it as input.
   a) Check, if the channel exists then googleapi-client.error sends the HTTP 200 code that says channel_id present in YouTube.
   b) If not, it sends the HTTP 400 code that says bad request and try with another channel_id.
4) Get the list of all video_id of a particular channel, along with the title and snippet of a particular video.
5) Get the comments of a particular video from input channel_id by the following steps.
   a) Check the HTTP code if it is 200 then retrieve the comments and store it in a list.
   b) Check the HTTP code if it is 400 then do nothing and follow the next channel_id. This situation occurs when the video is deleted by a channel member. So, if a video is deleted then all the comments associated with the video_id are also deleted.
   c) if 5.a and 5.b conditions are not satisfied then, it will raise the error called mismatch of video_id and terminate the program which is used to retrieve the comments.
6) The extracted comments are stored in a raw csv (comma-separated values) file called comments_raw csv file. This raw csv file contains bilingual sentences which are written in native scripts and roman scripts.

2) DATA PREPROCESSING

The channels selected from YouTube consist of roman scripts, native scripts and bilingual language sentences. On retrieval of all the raw comments, with the help of channel_id of a desired channel, only bilingual sentences (Malayalam - English) are pertinent in this context. All other sentences are ignored, driven by the primary aim of aggregating (Malayalam - English) code-mixed data set. In order to filter out code-mixed sentences from the retrieved sentences, a function is coded that has a regular expression, which identifies all the English alphabets, numbers, special characters, and emoticons. The filtered file eliminates sentences whose length (number of words) is less than six. Finally, we get the list of processed sentences, which is sent for data annotation. The following algorithm 1 illustrates how processed Malayalam-English code-mixed comments are retrieved.

Algorithm 1 To Convert Raw Comments to Processed Comments

Input: *comments_raw* ← comments/posts which are unprocessed both in native and roman script

Output: comments_processed ← Bilingual language (Malayalam-English)

```
contraction_map ← a mapping which maps some common contracted words to their complete form;
comments_processed ← Empty list;
//is_roman(sentence) returns true if comment do not contain any native script else false using a regular expression
//trim_repeated(word) trim consecutively repeated characters in the word
for comment in comments_raw do
    if is_roman(comment) then
        temp_comment ← Empty list;
        for word in comment do
            word ← trim_repeated(word);
            word ← contraction_map(word);
            temp_comment.append(word)
        end for
        comments_processed.append(temp_comment)
    end if
end for
save comments_processed to file;
```

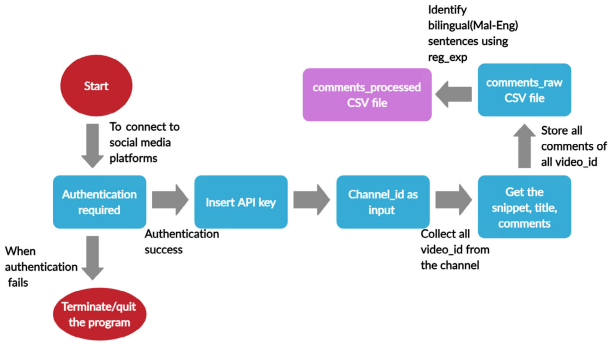

FIGURE 2. Data collection process starting from scraping YouTube comments to identifying bilingual Mal-Eng code-mixed text.

Fig. 2 presents an overall idea of data collection mechanism for Malayalam-English code-mixed data:

3) DATA ANNOTATION

After collection of the bilingual (Malayalam-English) sentences, data annotation process is initiated by introduction of the concepts like tokenization, which is a basic step of chopping sentences into piece of words, POS tagging which is the process of making corresponding POS tag for each tokenized word. Lemmatization is the process of removing inflectional endings of a word and returning the basic form of a particular dictionary word. Introduced the concept of lemmatization for POS_tag words like adverb, verb, adjective, noun. Then, some words are tagged as 'eng'

(English) by introducing a dictionary.[1] Tagged emoticons as 'univ' (universal) by deploying the regular expression from the emoji module in Python. Following this type of consequences steps, six tags are introduced: 'eng' (English), 'mal' (Malayalam),'univ' (universal), 'mix' (mixed), 'acr ' (acronyms), 'undef'(undefined) for the code-mixed data set. There are exceptions to this which is corrected manually. A detailed explanation of how tagging is performed is given in algorithm 2. A snapshot of the annotated data set is found in Table 3.

B. WORD LEVEL LANGUAGE IDENTIFICATION

In this subsection the major focus is on the models used for the prediction of language tags at the word level, and the evaluation metrics, used for verifying the performance of the models.

1) BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT)

The concept of transformers [53] was a major break-through in the domain of NLP. Transformers outperformed traditional neural networks like recurrent neural networks (RNN) [54] and convolutional neural networks (CNN) [55]. Let us take an example for illustration: ''Ritu, a data analyst, was pleased to learn the availability of a Data Scientist position at the Newtech Company. An ardent worker, she was responsible for big data projects at a Hydrogen company. She completed her Master of Science in Statistics, and has 10 years of experience in managing data related tasks. As a researcher in ML and Artificial Intelligence, she was conferred the young scientist award''. The underlined words refer to the same person - Ritu. As humans, we can easily correlate all the underlined words to the same person; but for a machine, detection of these relationships, over a long sequence of words in a sentence, is an intricate process.

Major advantage of transformer neural network architecture is all about parallelization of sequential data. It employs an encoder decoder architecture much like RNNs. The only difference is that the input sequence can be passed in parallel, in case of the transformer architecture. The architecture basically consists of encoder and decoder components. For example: His dog looks so cute. He looks like a dog. Here the word dog has different meanings in both the sentences. Hence, positional encoders are used, which is a vector that has information on distances between words in the sentence. After passing the sentence through the input embedding and application of positional encoding, we get word vectors which have positional information (contextual information). The contextual information vector is passed through the encoder block which has a multi-head attention layer and a feed-forward layer. The attention layer is a mechanism in which for every word, an attention vector is generated, which captures contextual relationships between words in the sentence. This is followed by the feed-forward neural network which is applied to all the attention vectors. It is used to transform the attention vectors for the next encoder or decoder block.

[1] https://inventwithpython.com/dictionary.txt

Algorithm 2 Word Level Comment Tagging

Input: *unlabeled_comments* ← File with processed comments/posts which are not tagged
Output: Word level tagged file
Initialization
eng_word_set ← Set of English words;
emoji_regex ← Regular Expression for emoticons;
stop_words ← English stop words from NLTK module;
tokenized_comments ← List of comments split by space;
tokenized_comments ← Remove comments with number of words less than 5 from tokenized_comments;
acronym_list ← Set of acronyms;
univ_exp_sym_dig ← Set of universal expression, symbols and digits;
annotated_comments ← Empty List;
//regex.match(word), returns true if the word matches the pattern else false
//word_length(word) returns the length of the word
//is_mixed(word) returns true if word is code-mixed else false

```
for comment in unlabeled_comments do
    temp_comments ← Empty List;
    for word in comment do
        if word in eng_word_set then
            temp_comments.append((word,"eng"))
        else if emoji_regex.match(word) then
            temp_comments.append((word,"univ"))
        else
            if word in stop_words then
                temp_comments.append((word,"eng"))
            else if word in acronym_list then
                temp_comments.append((word,"acr"))
            else if word inuniv_exp_sym_dig then
                temp_comments.append((word,"univ"))
            else if word_length(word)<3 then
                temp_comments.append((word,"undef"))
            else
                if word_length(word)>3 and
                is_mixed(word) then
                    temp_comments.append((word,"mix"))
                else
                    temp_comments.append((word,"mal"))
                end if
            end if
        end if
    end for
    annotated_comments.append(temp_comments)
end for
save annotated_comments to file;
```

TABLE 3. Snapshot of an annotated data set in WLLI.

Sentence ID	Comments
0	[('kidu', 'mal'), ('story', 'eng'), ('kidu', 'mal'), ('bgm', 'acr'), ('kidu', 'mal'), ('twist', 'eng'), ('kidu', 'mal'), ('kidu', 'mal'), ('kidu', 'mal'), ('thriller', 'eng'), ('oru', 'mal'), ('2', 'univ'), ('manikor', 'mal'), ('padm', 'mal'), ('tharun', 'mal'), ('same', 'eng'), ('feel', 'eng'), ('adipol', 'mal'), ('ithupola', 'mal'), ('film', 'eng'), ('inyum', 'mal'), ('cheyanm', 'mal')]
1	[('super', 'eng'), ('eniyum', 'mal'), ('kure', 'mal'), ('comedyu', 'mix'), ('ayi', 'mal')]
2	[('baviyl', 'mal'), ('oru', 'mal'), ('penu', 'mal'), ('verumbol', 'mal'), ('ithoke', 'mal'), ('avl', 'mal'), ('enkod', 'mal'), ('cheyipkum', 'mal'), ('ipo', 'mal'), ('then', 'eng'), ('cheytu', 'mal'), ('padikunth', 'mal'), ('nalth', 'mal')]

TABLE 4. Model type and model name.

Model_type	Model_name	Transformer Blocks (L)	Hidden Layer size (H)	Self-attention heads (A)	Number of Parameters
BERT	bert-base-uncased	12	768	12	110M
DistilBERT	distilbert-base-uncased	6	768	12	66M
ELECTRA	electra-base-discriminator	12	768	12	110M
XLM-RoBERTa	xlm-roberta-base	12	768	8	125M
CamemBERT	camembert-base	12	768	12	110M

In the decoder block the input embedding is used to get the vector form of the word, and add positional vectors, to get the context of a word in a sentence. This vector is passed to a decoder block which has 3 main components, two of which are similar to the encoder blocks. The self-attention block generates attention vectors for every word in the sentence. It shows how much each word is related to every word in the sentence. These attention vectors and vectors from the encoder are passed to the encoder decoder attention block. The output which is the attention vector is passed to feed-forward units, linear layer and SoftMax layer which is the probability distribution of words.

BERT is obtained by stacking the encoders of a transformer architecture. BERT is a bidirectional model, as it gains an understanding of a token's context from both its left and right sides, simultaneously, during the training phase. BERT is widely used in diverse applications of NLP like neural machine translation, question answers, sentiment analysis and text summarization. All these tasks require a clear understanding of the language. This model is trained on English data or on a combination of multiple languages. For each task, the BERT model can be fine-tuned by minimal changes to the training model. Hence, the training of BERT is done in two phases. The first phase is pre-training, where the model detects the language and context; the second phase is fine-tuning, where the model learns how to solve specific problems. The goal of pretraining is to make BERT learn the language and its context.

BERT variant architectures used in this paper are.

1) **XLM-RoBERTa:** XLM-RoBERTa [56] is a huge multi-lingual model, pretrained on a large amount of filtered data, that does not require special tensors to determine the language.
2) **CamemBERT:** CamemBERT [57] is based on Facebook's RoBERTa, but trained on French language. The camembert is fine-tuned on the developed code-mixed Malayalam-English language pair, along with the hyperparameter values. Since we had a substantial tag wise count, CamemBERT was able to learn with fine tuning. For fine tuning we take the 768 tokens from the last hidden state of CamemBERT and the embeddings of the 768 tokens are given to a full connected layer for classification of each token.
3) **DistilBERT:** DistilBERT [58] is a comparatively very small, quick, less expensive, and light-weight Transformer model, trained by distillation of Bert base.
4) **ELECTRA:** ELECTRA [59] pursues a pre-training methodology that trains 2 transformers: generator and discriminator. The generator replaces tokens in a sequence, and is trained as a masked model. The discriminator identifies which tokens in the sequence were initially replaced by the generator.

The overall architecture for the transformer based classification approach is given in Fig.3. Each token is a word; each of these words is converted into embeddings, using pretrained embeddings. Each word is given a sentence ID, so that the sentence bearing the context of each word can be preserved. The sentence IDs generated are incremented

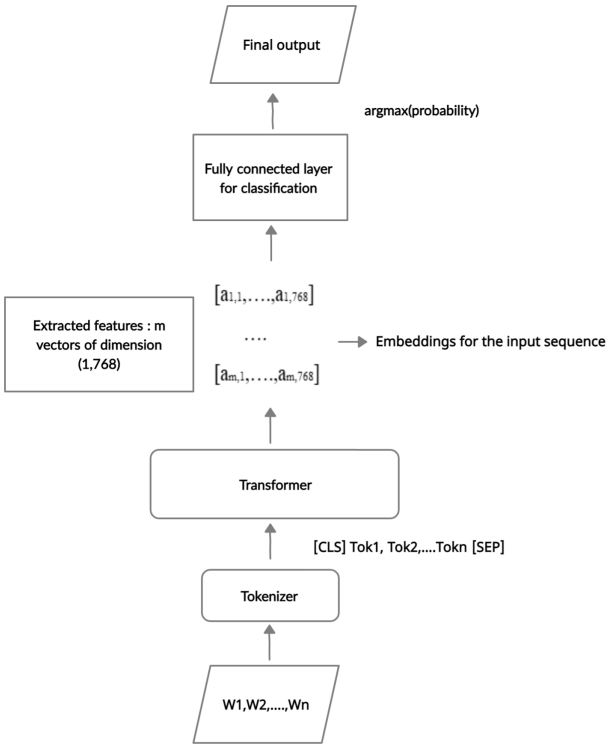

FIGURE 3. Overall architecture for the transformer based classification approach.

by 1 for each sentence. The input to the NERModel[2] is a list of lists, of the form [[sentence id, word, tag]]. The NERModel class was used for token (word) classification. To create a NERModel, we must specify model type and model name. The NERModel is internally calling tokenclassification model which takes the entire sentence as input and predicts labels for each word in the sentence. The exact architecture used for training the NERModel is shown in Table 4.

C. EVALUATION METRICS

The evaluation metrics used, and their definitions are noted here under:

- Accuracy which is the simple ratio of correct predictions to total number of predictions.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{1}$$

  where TP is True Positive, FP is False Positive, TN is True Negative, and FN is False Negative.
- Precision is the ratio of correct predictions to total positive predictions. High precision indicates low number of false positives.

$$Precision = \frac{(TP)}{(TP + FP)} \tag{2}$$

- Recall is the ratio of predicted positives to actual positives in a class.

$$Recall = \frac{(TP)}{(TP + FN)} \tag{3}$$

- F1-score which is the weighted mean of precision and recall. F1-score takes into consideration, both false-positive and false-negative observations.

$$F1 - score = 2 * (\frac{Precision * Recall}{Precision + Recall}) \tag{4}$$

  In an unbalanced data distribution, the F1 score gives a more reliable measure of goodness of the model.
- Confusion matrix is an $n \times n$ matrix that describes the performance of the model by computation of the misclassification rate. Graphically, the x-axis is represented as true labels and y-axis as predicted labels, or vice-versa. The diagonal of the matrix represents the correctness of the test data, signifying how well the predicted labels represent the true labels. The upper triangle and lower triangle of the matrix represents the misclassified samples with respect to each true class.

IV. VERIFICATION, RESULTS AND ANALYSIS

This section covers the evaluation and results of the various architectures of BERT models. A total of 50K code-mixed data sets were generated for doing WLLI. Since modern NLP is completely data-driven and relies largely on deep learning, the language tags predicted clearly indicate the advantage of using BERT and its variants. All the experiments and evaluations were run on Tesla K80 GPU, NVIDIA GPU driver version 418.67, CUDA version: 10.1 with 12 GB GPU memory. For evaluation of the BERT and its variant architectures, 80% of the code-mixed data was used for training, and the remaining 20% of the data was used for validation. The pre-trained BERT architecture was fine-tuned with four hyperparameters, to achieve statistically significant results.

- Epochs, which indicate the number of passes made through the entire data set transformer, were the hyperparameters used for reduction of the training error. Indiscriminate increase in the number of epochs may ensue in an overfitting problem, affecting the validation and performance of the model.
- Batch_size refers to the number of samples fed, at a time, into the transformer, in each iteration.
- Dropout, a regularization technique used to evade the overfitting problem, ensures generalization of the model. Creating multiple neural networks from an original neural network by just dropping some nodes.
- Learning rate is set so that the value of the loss function is minimized.

After an extensive tuning of hyperparameters the batch size is set to 16, learning rate to 2e-5, dropout at 0.1, epochs to be 10 and the maximum sequence length set to 128. Table 5 verifies that the ELECTRA model had the best performance for F1-score, out-performing CamemBERT by a marginal

[2]https://simpletransformers.ai/docs/ner-specifics

TABLE 5. Evaluation measures for transformer based word embedding methods.

Word Embedding Methods	Precision	Recall	F1-score	Accuracy
BERT	0.9934	0.9901	0.9917	0.9893
ELECTRA	**0.9937**	**0.9931**	**0.9933**	**0.9941**
CamemBERT	0.9793	0.9653	0.9718	0.9839
XLM-RoBERTa	0.9883	0.9811	0.9846	0.9899
DistilBERT	0.9893	0.9902	0.9897	0.9880

Note: The bold numbers represent the highest score obtained among evaluation matrices for the transformer based approach

TABLE 6. Tag-wise accuracy for transformer based classification models in WLLI.

Word Embedding Methods	acr	eng	mal	mix	undef	univ
BERT	0.9949	0.9848	0.9917	0.9649	0.9997	0.9998
ELECTRA	0.9974	0.9961	0.9915	0.9734	0.9997	0.9999
CamemBERT	0.9907	0.9902	0.9839	0.8305	0.9965	0.9996
XLM-RoBERTa	0.9902	0.9942	0.9884	0.9161	0.9976	0.9999
DistilBERT	0.9994	0.9826	0.9906	0.9687	0.9997	0.9998

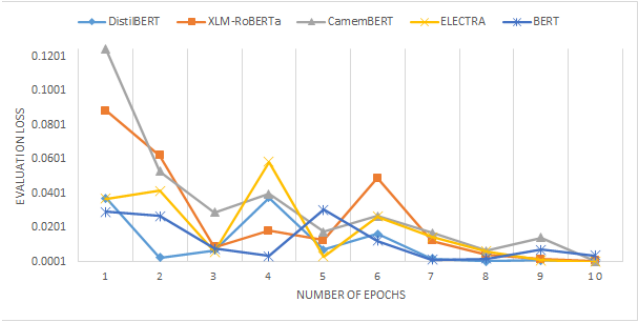

FIGURE 4. Relationship between number of epochs and the evaluation loss for the transformer based classification model.

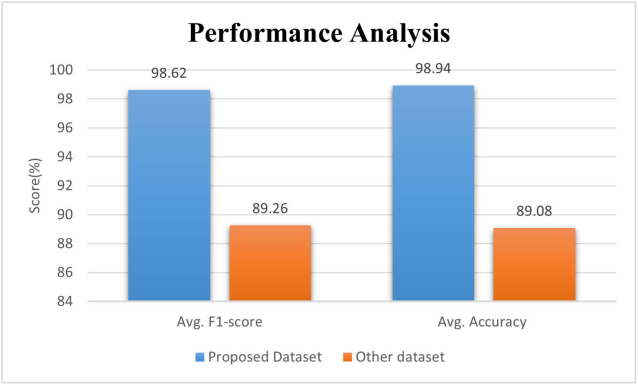

FIGURE 5. Validation of performance scores with Hi-En code-mixed data set.

value of 2.21% and BERT by a marginal value of 0.16%. DistilBERT was formed by distillation of the BERT model. It is a smaller, faster, cheaper and lighter transformer version of the BERT architecture. DistilBERT shows a marginal decrease of 0.36% from ELECTRA's F1 score. The most recently released XLM-RoBERTa model was trained on multiple languages over CommonCrawl, a large data set, whose size was 2.5 gigabytes. XLM-RoBERTa showed a 0.88% decrease from ELECTRA's F1 score. ELECTRA performed better than all the other pre-trained models for code-mixed Malayalam-English data set.

In Fig. 4, loss function is indicative of some corrections that need to be applied to a specific model. The loss curves, notably high at the get-go, after a few epochs follow steep descents, reaching a plateau with increased number of epochs; however, neither oscillations nor further decrease in loss was observed with additional increase in the number of epochs. The lower the evaluation loss, the higher is the precision, the recall and the F1-score, the better is the model. The evaluation loss is almost the same for all the five word embedding methods after 10 epochs. The decay in loss occurs

TABLE 7. Misclassification rate for various word embedding methods.

Word Embedding Methods	Values
BERT	0.0106
ELECTRA	0.0058
CamemBERT	0.0160
XLM-RoBERTa	0.0098
DistilBERT	0.0119

TABLE 8. Brief description of the labels and label distribution for Hi-En [60].

Label	Description	Hi-En (%)
lang1	English words alone	57.764
lang2	Hindi words alone	20.418
ne	Proper names	6.582
other	Symbols, usernames, emoticons	14.807
mixed	Mix of both English and Hindi	0.04
ambiguous	Not possible to determine whether English or Hindi	0.009
fw	Neither English nor Hindi	0.369
unk	Not recognizable word	0.012

TABLE 9. Performance scores obtained for Hi-En language pair using the proposed transformer based classification approach.

Word Embedding Methods	Precision	Recall	F1-score	Accuracy
BERT	0.9160	0.9032	0.9031	0.9031
ELECTRA	0.9048	0.9026	0.9006	0.9026
CamemBERT	0.8904	0.8789	0.8782	0.8789
XLM-RoBERTa	0.9060	0.8656	0.8774	0.8656
DistilBERT	0.9174	0.9047	0.9038	0.9047

faster for all the word embedding methods. The magnitude of the loss decreased from 0.1245 to 0.0001.

Table 6 illustrates the tag-wise accuracy of each model. The accuracy rate is highest for the 'univ' tags. ELECTRA and XLM-RoBERTa models achieved 0.9999 accuracy each, for the words with 'univ' tag. Some pre-trained models got confused while tagging the words that belonged to the 'mix' tag. The maximum accuracy achieved for this tag was 0.9734 for the ELECTRA model. Table 7, tabulated the misclassification rate for each model. It is evident that ELECTRA has the lowest misclassification rate of 0.005 due to highest accuracy, and CamemBERT, the highest misclassification rate of 0.016 due to lowest accuracy.

Fig. 6 shows the confusion matrices for the models pre-trained on the Malayalam-English code-mixed corpus. Confusion matrix was used to evaluate the performance of the output generated by the models. The x-axis represents the predicted labels while the y-axis represents the true labels. The diagonal elements present the majority of values for which the predicted and true labels are equal. The misclassification rate by the models is very small which signifies many correct predictions. In addition, the class-wise accuracy for each proposed word embedding techniques is outstanding. Interestingly, these models rarely confuse acronyms, undefined and universal labels, but misclassify English as Malayalam, or the other way around. Higher the number of points along the diagonal, better is the confusion matrix.

Next, the proffered approach was compared with code-mixed Hindi-English (Hi-En) language pair. Table 8 summarizes data set statistics for Hi-En code-mixed language pair. Furthermore, performance of the proposed method is compared, for Hi-En code-mixed data, and tabulated in Table 9. The transformer based classification method, which used variants of BERT, obtained the highest evaluation scores (precision, recall, F1- score and accuracy) when compared with Hi-En code-mixed language pair. Increase of 9% in both the F1-score score and accuracy validate the effectiveness of the novel approach of WLLI. The enhanced performance on 50 K code-mixed sentences is not anymore burdensome than fine-tuning pretrained BERT models.

Fig. 5 displays the performance analysis of the proposed Malayalam-English code-mixed data set over Hindi-English code-mixed data set. It is evident that the propounded data set outperformed evaluation scores due to more count per class as compared to the other benchmark data set. As SOTA transformer deep learning models are data hungry, they are enabled to correctly predict the classes, as a testimony to their notably high efficacy.

V. LIMITATIONS OF THE STUDY

To recapitulate, this paper pursued an audacious approach for the creation of an openly accessible code-mixed data set for the Malayalam-English Indian language pair. Close introspection of this maiden attempt revealed three probable areas that call for further investigation. Firstly, the study was centered on a transformer-based-classification scheme. This caveat calls for extensive evaluations with ML and DL (deep learning) approaches. Secondly in the developed 50 K code-mixed corpus with six labels, all the proper nouns are not distinctly tagged as named entities('ne'). Currently, they are placed under the 'mal' tag. These limitations are being worked on towards a fine-grained analysis of the propounded data set. Thirdly, this study was constrained by comments only from YouTube; a wider variety of comments, from other social media platforms, also need to be considered.

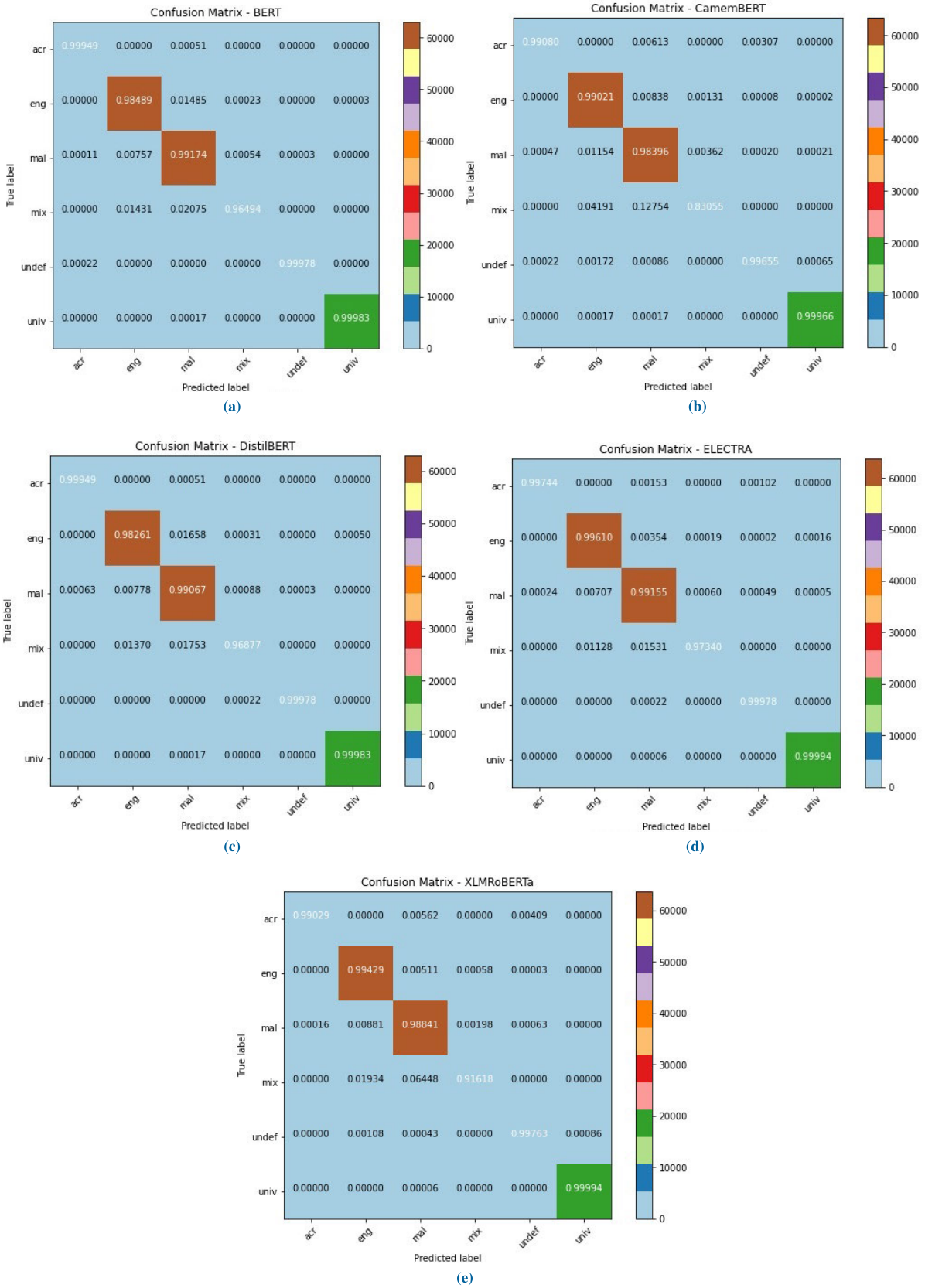

FIGURE 6. Confusion Matrices of (a) BERT (b) CamemBERT (c) DistilBERT (d) ELECTRA (e) XLMRoBERTa word embedding methods in WLLI.

VI. CONCLUSION AND FURTHER SCOPE

The current study is the first step towards developing a substantially large and rigorously code-mixed dataset for WLLI, for the Malayalam-English language pair. Language identification and classification are active areas of research in the code-mixed domain. They can potentially serve as a boon to the business community, by automated sentiment analysis over the demands of bilingual people. WLLI fulfills a major role in information retrieval tasks. Affronted by the daunting challenges for low-resource languages have been, there have been no gold standards available for Malayalam-English code-mixed data sets. In this paper, an innovative, WLLI - word-level language identification solution, was elucidated for assessments of Malayalam–English code-mixed text. The novel approach entails automatic generation and tagging of a bilingual data set into six labels. BERT, a context sensitive embedding method, along with its variants, formed the central concept of the proposed methodology. Fine-tuning of the pretrained BERT models enhanced the performance on 50K code-mixed sentences. The verification results validated the capability of these multilingual models to tackle such problems, even with a fewer number of epochs on the generated data set. Comparative evaluations of these models affirmed that ELECTRA gave a marginal rise of 1.03% in accuracy and 2.21% in F1-score when compared with its other variants. ELECTRA obtained the highest evaluation metrics - F1-score of 0.9933, accuracy- 0.9941, precision - 0.9937, and 0.9931 for recall. ELECTRA had a minimal misclassification rate of 0.005 and the lowest reported evaluation loss of 0.0001, over 10 epochs. A majority of the tokens belonged to English and Malayalam tags, which is quite natural for data sets in a real world scenario; a smaller number of tokens belong to other tags, leading to class imbalance problems. This ensues in a low misclassification rate for the tag, which contains a proportionately higher number of samples; hence, most of the test samples get misclassified to the majority tag. The proffered method could be improved by data augmentation techniques, besides including data from other popular media platforms - Facebook, Twitter, and WhatsApp.

REFERENCES

[1] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[2] A. K. Joshi, "Processing of sentences with intra-sentential code-switching," in *Proc. 9th Conf. Comput. Linguistics*, 1982, pp. 1–6.

[3] K. Shanmugalingam, S. Sumathipala, and C. Premachandra, "Word level language identification of code mixing text in social media using NLP," in *Proc. 3rd Int. Conf. Inf. Technol. Res. (ICITR)*, Dec. 2018, pp. 1–5.

[4] K. R. Mabokela, M. J. Manamela, and M. Manaileng, "Modeling code-switching speech on under-resourced languages for language identification," in *Proc. SLTU, ISCA*, Russia, 2014, pp. 225–230.

[5] U. Barman, A. Das, J. Wagner, and J. Foster, "Code mixing: A challenge for language identification in the language of social media," in *Proc. 1st Workshop Comput. Approaches Code Switching*, 2014, pp. 13–23.

[6] C. I. Eke, A. A. Norman, and L. Shuib, "Context-based feature technique for sarcasm identification in benchmark datasets using deep learning and BERT model," *IEEE Access*, vol. 9, pp. 48501–48518, 2021.

[7] J. Milroy, *One Speaker, Two Languages: Cross-Disciplinary Perspectives on Code-Switching*, vol. 10. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[8] P. Auer, *Code-Switching in Conversation: Language, Interaction and Identity*. Evanston, IL, USA: Routledge, 2013.

[9] C. R. Voss, S. Tratz, J. Laoudi, and D. M. Briesch, "Finding Romanized Arabic dialect in code-mixed tweets," in *Proc. LREC*, 2014, pp. 2249–2253.

[10] H. Elfardy, M. Al-Badrashiny, and M. Diab, "Code switch point detection in Arabic," in *Proc. 18th Int. Conf. Appl. Natural Lang. Inf. Syst. (NLDB)*, MediaCity, U.K., Jun. 2013.

[11] Y. Li, Y. Yu, and P. Fung, "A mandarin-english code-switching corpus," in *Proc. LREC*, vol. 2012, pp. 2515–2519.

[12] A. Das and B. Gambäck, "Identifying languages at the word level in code-mixed Indian social media text," in *Proc. 11th Int. Conf. Natural Lang. Process.*, 2014, pp. 378–387.

[13] D. Nguyen and A. S. Doğruöz, "Word level language identification in online multilingual communication," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 857–862.

[14] K. C. Raghavi, M. K. Chinnakotla, and M. Shrivastava, "'Answer Ka type Kya he?': Learning to classify questions in code-mixed language," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 853–858.

[15] N. Jain and R. A. Bhat, "Language identification in code-switching scenario," in *Proc. 1st Workshop Comput. Approaches Code Switching*, 2014, pp. 87–93.

[16] L.-C. Yu, W.-C. He, W.-N. Chien, and Y.-H. Tseng, "Identification of code-switched sentences and words using language modeling approaches," *Math. Problems Eng.*, vol. 2013, pp. 1–7, Jan. 2013.

[17] G. Molina, F. AlGhamdi, M. Ghoneim, A. Hawwari, N. Rey-Villamizar, M. Diab, and T. Solorio, "Overview for the second shared task on language identification in code-switched data," in *Proc. 2nd Workshop Comput. Approaches Code Switching*, 2016, pp. 40–49. [Online]. Available: https://aclanthology.org/W16-5805

[18] S. Banerjee, N. Moghe, S. Arora, and M. M. Khapra, "A dataset for building code-mixed goal oriented conversation systems," in *Proc. 27th Int. Conf. Comput. Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 3766–3780. [Online]. Available: https://aclanthology.org/C18-1319

[19] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, "YAKE! Keyword extraction from single documents using multiple local features," *Inf. Sci.*, vol. 509, pp. 257–289, Jan. 2020.

[20] D. Jain, A. D. Prabhu, S. Vatsal, G. Ramena, and N. Purre, "Codeswitched sentence creation using dependency parsing," in *Proc. IEEE 15th Int. Conf. Semantic Comput. (ICSC)*, Jan. 2021, pp. 124–129.

[21] D. Gupta, A. Ekbal, and P. Bhattacharyya, "A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning," in *Proc. Findings Assoc. Comput. Linguistics (EMNLP)*, 2020, pp. 2267–2280. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.206

[22] V. Srivastava and M. Singh, "Challenges and limitations with the metrics measuring the complexity of code-mixed text," in *Proc. 5th Workshop Comput. Approaches Linguistic Code-Switching*, 2021, pp. 6–14. [Online]. Available: https://aclanthology.org/2021.calcs-1.2

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805

[24] N. B. Sristy, N. S. Krishna, B. S. Krishna, and V. Ravi, "Language identification in mixed script," in *Proc. 9th Annu. Meeting Forum Inf. Retr. Eval.*, Dec. 2017, pp. 14–20.

[25] G. Chittaranjan, Y. Vyas, K. Bali, and M. Choudhury, "Word-level language identification using CRF: Code-switching shared task report of MSR India system," in *Proc. 1st Workshop Comput. Approaches Code Switching*, 2014, pp. 73–79.

[26] P. Lamabam and K. Chakma, "A language identification system for code-mixed English-Manipuri social media text," in *Proc. IEEE Int. Conf. Eng. Technol. (ICETECH)*, Mar. 2016, pp. 79–83.

[27] P. V. Veena, M. A. Kumar, and K. P. Soman, "An effective way of word-level language identification for code-mixed Facebook comments using word-embedding via character-embedding," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1552–1556.

[28] T. Solorio and Y. Liu, "Part-of-speech tagging for English-Spanish code-switched text," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2008, pp. 1051–1060.

[29] M. Rosner and P. J. Farrugia, "A tagging algorithm for mixed language identification in a noisy domain," in *Proc. Interspeech*, Antwerp, Belguim, Aug. 2007, pp. 190–193.
[30] P. V. Veena, M. A. Kumar, and K. P. Soman, "Character embedding for language identification in Hindi-English code-mixed social media text," *Computación y Sistemas*, vol. 22, no. 1, pp. 65–74, Mar. 2018.
[31] R. Sequiera, M. Choudhury, P. Gupta, P. Rosso, S. Kumar, S. Banerjee, S. K. Naskar, S. Bandyopadhyay, G. Chittaranjan, A. Das, and K. Chakma, "Overview of fire-2015 shared task on mixed script information retrieval," in *Proc. FIRE Workshops*, vol. 1587, 2015, pp. 19–25.
[32] N. Bansal, V. Goyal, and S. Rani, "Experimenting language identification for sentiment analysis of English Punjabi code mixed social media text," *Int. J. E-Adoption*, vol. 12, no. 1, pp. 52–62, Jan. 2020.
[33] G. A. Guzman, J. Serigos, B. E. Bullock, and A. J. Toribio, "Simple tools for exploring variation in code-switching for linguists," in *Proc. 2nd Workshop Comput. Approaches Code Switching*, 2016, pp. 12–20.
[34] W. Adouane and S. Dobnik, "Identification of languages in Algerian Arabic multilingual documents," in *Proc. 3rd Arabic Natural Lang. Process. Workshop*, 2017, pp. 1–8.
[35] S. Shekhar, D. K. Sharma, and M. M. S. Beg, "Language identification framework in code-mixed social media text based on quantum LSTM—The word belongs to which language?" *Modern Phys. Lett. B*, vol. 34, no. 6, Feb. 2020, Art. no. 2050086.
[36] N. Sarma, S. R. Singh, and D. Goswami, "Word level language identification in Assamese-Bengali-Hindi-English code-mixed social media text," in *Proc. Int. Conf. Asian Lang. Process. (IALP)*, Nov. 2018, pp. 261–266.
[37] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas, "'I am borrowing ya mixing?" an analysis of English-Hindi code mixing in Facebook," in *Proc. 1st Workshop Comput. Approaches to Code Switching*, 2014, pp. 116–126.
[38] L. King, E. Baucom, T. Gilmanov, S. Kübler, D. Whyatt, W. Maier, and P. Rodrigues, "The IUCL+ system: Word-level language identification via extended Markov models," in *Proc. 1st Workshop Comput. Approaches Code Switching*, 2014, pp. 102–106.
[39] A. Jaech, G. Mulcaire, M. Ostendorf, and N. A. Smith, "A neural model for language identification in code-switched tweets," in *Proc. 2nd Workshop Comput. Approaches Code Switching*, 2016, pp. 60–64.
[40] S. Mandal and A. K. Singh, "Language identification in code-mixed data using multichannel neural networks and context capture," 2018, *arXiv:1808.07118*. [Online]. Available: http://arxiv.org/abs/1808.07118
[41] A. Jamatia, A. Das, and B. Gambäck, "Deep learning-based language identification in English-Hindi-Bengali code-mixed social media corpora," *J. Intell. Syst.*, vol. 28, no. 3, pp. 399–408, Jul. 2019.
[42] K. Singh, I. Sen, and P. Kumaraguru, "Language identification and named entity recognition in hinglish code mixed tweets," in *Proc. ACL Student Res. Workshop*, 2018, pp. 52–58.
[43] R. Marković, M. Gosak, M. Perc, M. Marhl, and V. Grubelnik, "Applying network theory to fables: Complexity in slovene belles-lettres for different age groups," *J. Complex Netw.*, vol. 7, no. 1, pp. 114–127, Feb. 2019.
[44] Y. Samih, S. Maharjan, M. Attia, L. Kallmeyer, and T. Solorio, "Multilingual code-switching identification via LSTM recurrent neural networks," in *Proc. 2nd Workshop Comput. Approaches Code Switching*, 2016, pp. 50–59.
[45] T. Ranasinghe, M. Zampieri, and H. Hettiarachchi, "BRUMS at HASOC 2019: Deep learning models for multilingual hate speech and offensive language identification," in *Proc. FIRE Working Notes*, 2019, pp. 199–207.
[46] P. Patwa, G. Aguilar, S. Kar, S. Pandey, S. PYKL, B. Gambäck, T. Chakraborty, T. Solorio, and A. Das, "SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets," in *Proc. 14th Workshop Semantic Eval.*, Dec. 2020, pp. 774–790. [Online]. Available: https://aclanthology.org/2020.semeval-1.100
[47] S. Thara, E. Sampath, P. Reddy, "Code mixed question answering challenge using deep learning methods," in *Proc. 5th Int. Conf. Commun. Electron. Syst. (ICCES)*, Jun. 2020, pp. 1331–1337.
[48] K. Asnani and J. D. Pawar, "Automatic aspect extraction using lexical semantic knowledge in code-mixed context," *Procedia Comput. Sci.*, vol. 112, pp. 693–702, Jan. 2017.
[49] K. Abdalgader and A. A. Shibli, "Experimental results on customer reviews using lexicon-based word polarity identification method," *IEEE Access*, vol. 8, pp. 179955–179969, 2020.
[50] J. Kim, Y. Ko, and J. Seo, "Construction of machine-labeled data for improving named entity recognition by transfer learning," *IEEE Access*, vol. 8, pp. 59684–59693, 2020.
[51] E. Yılmaz, H. van den Heuvel, and D. van Leeuwen, "Investigating bilingual deep neural networks for automatic recognition of code-switching frisian speech," *Procedia Comput. Sci.*, vol. 81, pp. 159–166, Jan. 2016.
[52] I. Bhat, R. A. Bhat, M. Shrivastava, and D. Sharma, "Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics, Short Papers*, vol. 2, 2017, pp. 324–330. [Online]. Available: https://aclanthology.org/E17-2052
[53] T. Wolf, J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer, and R. Louf, "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*. Stroudsburg, PA, USA: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6
[54] K. Soman, "Amrita-CEN@ MSIR-FIRE2016: Code-mixed question classification using bows and RNN embeddings," in *Proc. FIRE Working Notes*, 2016, pp. 122–125.
[55] D. Jain, A. Kumar, and G. Garg, "Sarcasm detection in mash-up language using soft-attention based bi-directional LSTM and feature-rich CNN," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106198.
[56] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," 2019, *arXiv:1911.02116*. [Online]. Available: https://arxiv.org/abs/1911.02116
[57] L. Martin, B. Müller, P. J. O. Suárez, Y. Dupont, L. Romary, E. V. de la Clergerie, D. Seddah, and B. Sagot, "Camembert: A tasty French language model," 2019, *arXiv:1911.03894*. [Online]. Available: https://arxiv.org/abs/1911.03894
[58] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*. [Online]. Available: https://arxiv.org/abs/1910.01108
[59] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," 2020, *arXiv:2003.10555*. [Online]. Available:https://arxiv.org/abs/2003.10555
[60] D. Mave, S. Maharjan, and T. Solorio, "Language identification and analysis of code-switched social media text," in *Proc. 3rd Workshop Comput. Approaches Linguistic Code-Switching*, 2018, pp. 51–61.

S. THARA received the B.Tech. degree in computer science engineering from Cochin University of Science and Technology (CUSAT), in 2011, and the M.Tech. degree in computational engineering and networking from Amrita Vishwa Vidyapeetham, India, in 2013, where she is currently pursuing the Ph.D. degree in natural language processing/text mining/information extraction. She is also working as an Assistant Professor with the Department of Computer Science and Engineering, Amrita School of Engineering, Amritapuri, Kerala, India. Her research interests include natural language processing, social media text analytics, machine learning, and deep learning.

PRABAHARAN POORNACHANDRAN is currently a Professor with Amrita Vishwa Vidyapeetham. He has more than two decades of experience in computer science and security areas. His publication topics include pattern classification, text analysis, Internet, invasive software, recurrent neural nets, security of data, telecommunication traffic, computer network security, natural language processing, neural nets, time series, computer crime, convolution, cryptography, data mining, diseases, feedforward neural nets, multilayer perceptrons, neural net architecture, neurophysiology, social networking (online), telecommunication computing, vibrations, and application program interfaces. His research interests include AI, machine learning, malware, critical infrastructure security, and complex binary analysis.