Code Mixing “computationally bahut challenging hai”

Comprehensive Viva Report

Prashant Kodali - PhD CSE - 2018801011

August 2021

Contents

1 Introduction

Code mixing and Code switching refer to the phenomena where in a multilingual person will alternate between two or more languages. Code Switching is "juxtaposition within the same speech exchange of passages of speech belonging to two different grammatical systems or subsystems" [1]. Code mixing refer to all cases where lexical items and grammatical features from two languages appear in one sentence [2]. In language contact literature [2, 3, 4], sometimes researchers have distinguished between the terms Code mixing and Code switching. In terms of the definition,

- **Code mixing** refers to the mixing of various linguistic units (morphemes, words, modifiers, phrases, clauses and sentences) primarily from two participating grammatical systems within a sentence.
- **Code switching** refer to the use of various linguistic units (words, phrases, clauses, and sentences) primarily from two participating grammatical systems across sentence boundaries within a speech event.

Usually, Code Switching is inter-sentences while Code mixing is intra-sentential phenomena. The distinction between Code mixing and Code Switching is controversial, and there are differing perspective among scholars on the specific definition of the two terms, and some have questioned the utility of maintaining such distinction[3].

In this report, we discuss state of current computational pipelines for processing sentences (as part of larger speech act, or on its own) which have entities from more than one language. Irrespective of the nature of mixing two languages: word borrowing, Code mixing, Code switching - computationally, all these manifestations pose a challenge for automatic text processing pipelines. Hence, we don't make the distinction between Code mixing, Code switching. We have used these terms, Code mixing/switching interchangeably through out this report.

In the following chapters, we attempt to understand the existing computational methodologies (data, pipelines etc) proposed for handling Code mix sentences, and identify aspects which can be improved for enhancing the performance of automatic processing of Code mix sentences.

To quote a few examples of Code mixing :

1. **Language Pair**: Hindi-English
   Example: Hum khelne jaa raha hain. Why don't you join us?
   Translation: We are going to play. Why don't you join us?
2. **Language Pair**: Malyalam-English
   Example: nee naale evng shoppingnu varunnundo?[5]
   Translation Are you coming for shopping tomorrow evening?
3. **Language Pair**: Bengali-English
   Example: Yaar tu to, GOD hain. tui JU te ki korchis? Hail u man! [6]

Translation: Buddy you are GOD. What are you doing in JU? Hail u man!

4. **Language Pair**: Hindi-English
   Example: Main kal movie dekhne jaa rahithi and raaste me I met Sudha. [7]
   Translation : I was going for a movie yesterday and on the way I met Sudha.

5. **Language Pair**: German-Turkish
   Example: Frau Kummer. Echte Name-**si** Christa. [8]
   Translation: 'Ms. Kummer. (Her) real name is Christa'

Example (1) demonstrates inter-sentential Code mixing between english and hindi , where first part of the example is in romanized Hindi, and the second part is in English. Example (2,5) are examples of intra-sentential Code mixing involving English - Malyalam, and Turkish-German, respectively. Example (2) exhibits two additional properties: a) Spelling of "evening" ; b) the inflection of the word "shopping" to "shoppingnu", which is accusative case for the word "shopping" in Malyalam. Similarly, in Example (5) a Turkish possessive case marker (-si) is attached to a German noun. Example (3) showcases Code mixing involving English Hindi and Bangla, and is a mixture of inter-sentential and intra-sentential Code mixing. Example (4) demonstrates intra-sentential Code mixing in Hindi and English. Both inter-sentential and intra-sentential Code mixing occurrences pose various challenges to the computational tools used to automatically process text. Various linguistic studies have made distinction between Code mixing and Code switching based on certain criteria, while others have used either of these terms as an umbrella term to represent any kind of language mixing. In this report, Code mixing and Code switch are used as umbrella terms for all types of mixing and are used interchangeably, as all of these phenomena pose challenges, of varying degrees, for computational systems used to automatically process such text.

1.1 Utility of Code mixing

Twitter users are known to generate multilingual texts (Ling et al., 2013, 2014), with Rijhwani et al. (2017) estimating that 3.5% of tweets are Code switched [9, 10, 11]. Further, users also modulate their writing style for different social registers (Eisenstein, 2015; Tatman, 2015).

It may also be motivated by social, psychological and conversational factors. When bilinguals switch or mix two languages, there might be underlying reasons causing that particular way of Code switching and Code mixing. [12] lists some of the reasons and motivations behind Code mixing. There are number of factors such as the participants in a conversation, the topic of conversation, when and where such conversation is occurring,

- Participant Roles and Relationship : more likely to Code mix with friends that with parents.
- Situational Factors : a particular language is more suited to particular group or topic.
- Message-Intrinsic Factors : Usage of Code mixing when quoting, reiteration, idioms , interjections etc.

- Language Attitudes, Dominance, and Security : more likely to use mix elements of dominant language when using non-dominant language,
- Bilinguals' Perception of Code Mixing and Code switching : Used in interpersonal , informal settings and interactions.

In [13], authors analyse the Hi-En Code mix tweets from different domains to identify the pragmatic function of Code mixing and propose a annotation methodology for the same. Tweets were collected from different domains (Sports, Movies, Politics etc.) and classify them into different languages : English, Romanised Hindi, Code Switching and others. Each tweet is annotated with their pragmatic functional category as mentioned below :

1. Narrative - Evaluative : Tendency to switch when moving from expressing facts to opinions.
   Example : petrol prices up by rs 3.18/litre, diesel by rs 3.09/litre. sab ki aesi tesi kr di.
2. Reinforcement : Reinforcing a opinion by a related one.
   Example : best wishes to indian team tiranga aapke saath hai
3. Sarcasm: Switch language to express a sarcastic opinion.
   Example : all is good...but paisa kahase aayega prabhu
4. Quotations: Quoting a sentence from a language and expressing opinion about it in another language. Example : 'bhaag modi bhaag' will be a national slogan very soon!
5. Imperative: Switching between an opinion and imperative statement.
   Example : please stop this aapstorm mein ek aam kisaan hu aur meri fasal kharab ho jayegi
6. Cause - Effect: Switch between reason or cause for a statement.
   Example : no need to worry bade bade matches main choti choti galtiyan hoti rehti hai indvssa
7. Reported Speech : To quote a sentence from another conversation.
   Example : drkumarvishwas had said during victory celebration after anna fast that: janlokpal pass hone do wo jashn hoga duniya dekhegi.
8. Abuse / Negative Sentiment : Switching to express a negative sentiment.
   Example : Seeing the movie I thought ki kisi bandar ke haath me camera de do to wo bhi movie banaa le
9. Others : variety of other reasons like Wishing, greetings etc.
   Example : good morning ...aaj ka din kitna achhaa hai...aisa lag raha sapna dekh rahe hai

There might be more switching categories if we look at other social media, texts other than social media and speech data.

[7] analyse Code mix sentences from Facebeook generated by En-Hi bilingual users and

There isn't hard boundary between lexical borrowing and Code mixing. The words first manifest as borrowed Code mixing and then slowly gets adapted into the lexicon of the language by repeated usage;

2 Typological Frameworks for Code mixing

1. Does Code mixing have a "third-grammar" of it's own? Since grammar is relevant at a sentence level, this line of enquiry will address only the intra-sentential Code mixing.

2. Is there a grammatical theory/generative grammar for Code mixing which can generate all legitimate Code mix sentences ?

3. Are the grammatical theories language specific or are localised to a particular language pair?

4. Do the questions listed above have any implication on computational approaches to Code mixing?

We try to address these questions in this chapter.

2.1 Difference Between Word Borrowing and Code mixing

[14] mention that "It is impossible in principle and in practice to draw an absolute boundary between Code switching and borrowing. They are indeed two separate phenomena, but they are linked by a continuum: as in so many other areas of historical linguistics, the dividing line between them is fuzzy, not sharp". The same has been reinforced by [7], who say that Code switched word or other morpheme becomes a borrowing as its usage becomes more frequent, until it is a assimilated as part of the recipient language, learned as such by new learners.

In [15], authors suggest that linguists define broadly three forms of borrowing, (i) cultural, (ii) core, and (iii) therapeutic borrowings. In cultural borrowing, a foreign word gets borrowed into native language to fill a lexical gap. This is because there is no equivalent native language word present to represent the same foreign word concept. For instance, the English word 'computer' has been borrowed in many Indian languages since it does not have a corresponding term in those languages3 . In core borrowing, on the other hand, a foreign word replaces its native language translation in the native language vocabulary. This occurs due to overwhelming use of the foreign word over native language translation as a matter of prestige, ease of use etc. For example, the English word 'school'.

2.2 Constraint Based Grammatical Theories for Code mixing

Early research in the field of Code mixing identified that, like any other linguistic phenomena, even Code mixing is constrained and rule-governed, and the analysis that followed this observation attempted to formulate language specific constraints that answer some of the aforementioned questions

[16]. Various constraint based theories have been proposed to explain various patterns of Code mixing instances.

Linguistic analysis into Code mixing and its typology has a long tradition, and over the years there have been various constraint based theories proposed to explain Code mixing patterns. [2] provides a framework to categorize the various constraints proposed into distinct buckets and highlight interplay between these abstracted categories of Code mix theories. Following is a list of the most prominent ones:

- **Insertion**: lexical units from one language are inserted into a structure of another language. Insertion of an alien lexical or phrasal category into a given structure. Approaches building on the notion of "insertion" attempt to formulate constraints in terms of structural properties of some base or matrix structure. Difference in approaches could be of size/type of inserted element : noun versus noun phrase.

  One such example could be Matrix language framework (MLF) proposed by [17]. MLF differentiates between matrix language (ML) and the embedded language (EL). Matrix language provide the syntactic framework for the utterance, dictating the position of content and functional words, and content words from Matrix or Embedded Language can be placed in the syntactic framework.

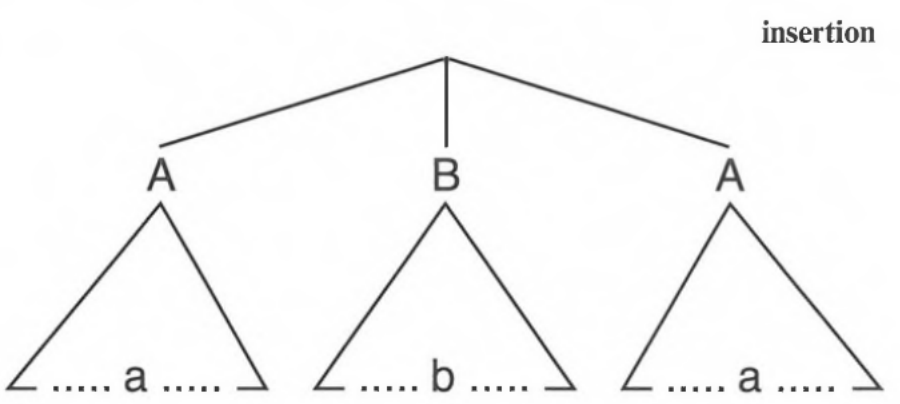


Figure 1: Code mixing based on Insertion process : Structurally alteration would be akin to the figure above. Here A and B are language labels for non-terminal nodes (markers of constituents belonging to one language), and a,b are labels for terminal nodes, indicating that the words chosen are from particular language. In this figure single constituent B (with words b from the same language) is inserted into a structure defined by language A, with words a from that language. [2]

- **Alteration**: Alteration between structures from languages. Grammatical constraints proposed based on these process, view constraints on mixing in terms of compatibility or equivalence of langauges involved at the switch point. An example of constraint based on this principle is Poplack's Equivalence Constraint and Free Morpheme Constraint which proposes that
  - Equivalence Constraint: Code mixing occurs largely at sites of equivalent constituent order.

Codes will tend to be switched at points where the surface structures of the languages map onto each other. Code switches are allowed within constituents so long as the word-order requirements of both languages are met at sentential level.

– Free Morpheme constraint : A switch may not occur between a bound morpheme and a lexical item unless the latter has been phonologically integrated into the language of the bound morpheme.

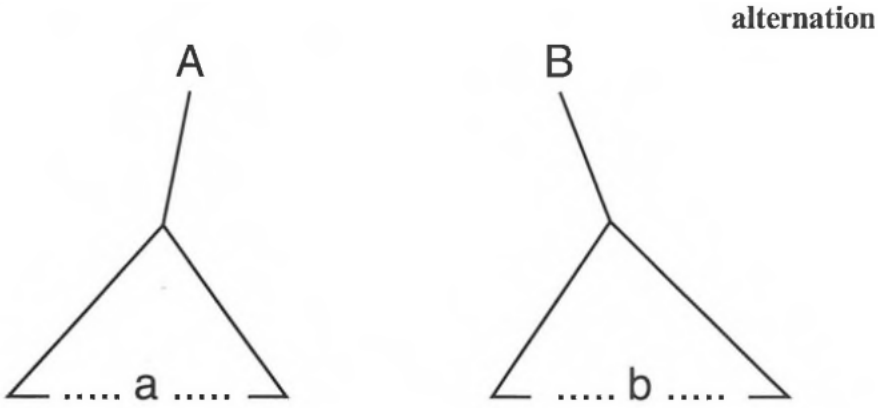


Figure 2: Code mixing based on Alteration process : Here a constituent from language A (with words from the same language) is followed by a constituent from language B (with words from that language). The language of the constituent dominating A and B is unspecified.[2]

- **Congruent Lexicalization**: Congruent Lexicalisation is assimilation of material from different lexical inventories into a shared grammatical structure. Congruent Lexicalisation hypothesises that there is a largely shared structure, lexicalised by elements from either language. The notion of congruent lexicalization underlies the study of style shifting and dialect/standard variation, rather than bilingual language use proper, and addresses the languages contact and assimilation aspect of multilingualism.

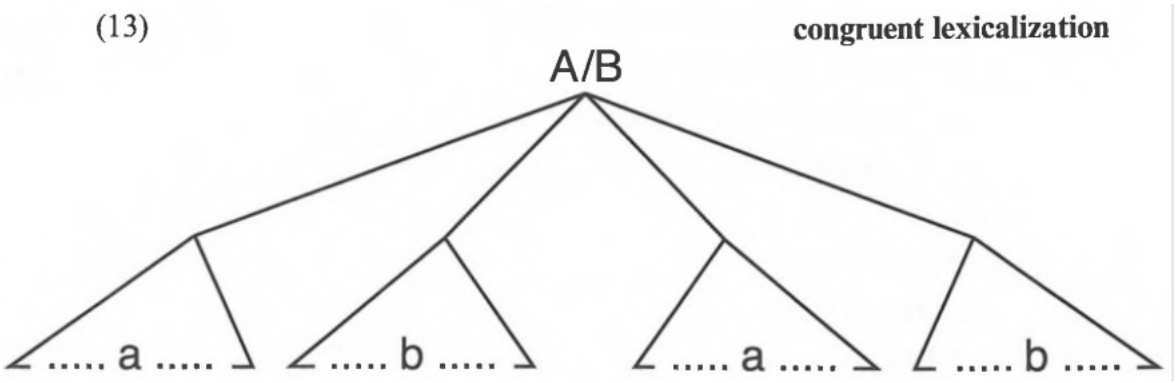


Figure 3: Code mixing based on Congruent Lexicalisation process : Here the grammatical structure is shared by languages A and B, and words from b oth languages a and b are inserted more or less randomly. [2]

The difference between these three processes as formulated by [2] is gradual and not a absolute. This implies particularly that in many immigrant communities, insertion o f new items and expressions into the home language can evolve into congruent lexicalization and then possibly into alternation (with set phrases and expressions from the ethnic language interspersed in the new language).

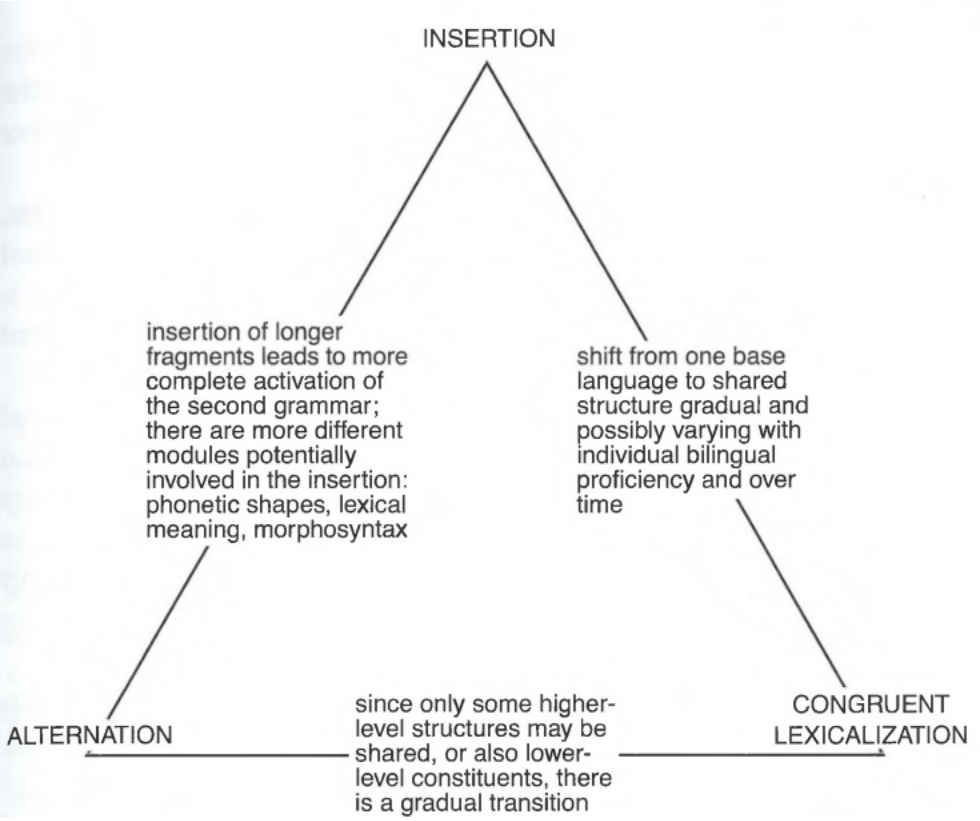


Figure 4: Schematic representation of the three main styles of Code mixing and transitions between them.[2]

However, for all the proposed constraints across aformentioned categories, there have been multiple counter-examples and objections made. [16] demonstrated that an approach which engages in the analysis of mix-language data, without reference to specifically crafted constraints or other mechanisms, can be successful, and may reveal interesting and subtle properties of the languages under analysis. These theories provide a structured framework for analysis and building automatic computational pipelines for Code mix sentences. Some of these syntactic theories of Code mixing have under-pined computational tools : [18] demonstrated a toolkit for automatically generate Code mixed data given parallel data in two languages, using Equivalence Constraint theory and Matrix language theory. While linguistics have focused on formulating the topological framework for Code mixing, NLP research has largely, with a few exceptions, relied on the notion of matrix language model that was advanced by [19, 17]. There have been cases where incorporating grammatical theories into computational work improved the performance of the models [8], hence combining theoretical understanding of

Code mixing into computational tools could create better tools for processing Code mixed language.

3 Challenges In Processing Code Mix Text

3.1 Data Availability and Collection

Sociolinguists have linked presence of Code mixing, or lack thereof, to the situational setting in which communication is taking place, where in Code mixing is "restricted to the role of informal communication in private settings, while the more prestigious cosmopolitan language is considered the voice of intellect and of public formal communication" [20]. Code mixing occurs more in informal settings. Thus Code mixing is more likely to seen in speech corpora and in social media data. However, both present their challenges: privacy of the speaker. Speech recording needs prior permissions, and sharing social media has its own privacy concerns. Another challenge is collecting such data. There are following options one can consider :

- **Collect data around a topic and then filter out the Code mix instances**
  As an example, we can consider collecting data from online social networks, using hashtags or query terms that are are trending in a state or city, and then try to filter out instances which are Code mix using a Language identification or such heuristics. However, the primary problem with such approach is that they end up casting a wide net : collecting a lot of data out of which only a few turn out to be Code mix.
- **Collect data using query terms which are more likely to have Code mix sentences**
  As an example, we can collect data using query terms such as: " happy ki app" or "i wish ki", which are more likely to collect instances of Code mix data. Compared to the method described earlier, this is more likely to have a higher percentage of Code mixed sentences. However, the query terms can make the collected corpora susceptible to selection bias. Methodology of finding and selecting such high-Code mix-yield becomes highly crucial step.
- **Identifying users on online social networks who frequencty Code mix and collect their posts**
  In [21], instead of use specific query words, authors identified Twitter user profiles who were frequently Code mixing and used their timelines to build a Code mixed corpus. But finding the initial set of usernames who are frequently Code mixing is difficult.

.

3.2 Source of Code Mix Data

Getting Code mix data is a challenge : because it is more likely to occur in speech events, informal settings. Early research of Code mixing was carried out on the spoken communication between

multilingual speakers. But with the advent of Internet and with increasing usage of Internet, the availability of Code mixing in written form gets more accessible. As the usage of Internet has expanded there has been a evolution of language contact, where people interact with others in their own native language or a mixture of languages if they are multilingual. Further, Internet also provides the informal setting where users are more likely to interact using native language or mixture of languages. There have been various studies which have looked at Code mixing on various social media platforms. Some have looked at it from a computational perspective for automatic processing of such text and some from the linguistic perspective to identify the nature of Code mixing in written form.

3.2.1 Code Mixing in Social Media

[22] and [23] analyse the online Usenet news forums used by emigre sikh communities which are multilingual, Internet Relay Chats (IRC), where bilingual users are active. Their analysis shows that the more Code mixing is found in synchronous IRC (where two people are simultaneously online) than in asynchronous Usenet forums, thereby postulating that the tendency for synchronous modes to favor Code mixing. They also observe that although English remains the major language for interaction, there are substantial number of instances of interactions using native language and a mix of English and native language. [24] present analysis of script-mixing between En-Hi languages in Twitter, identifying various reasons behind mixing scripts of two languages, and showing few examples of Code mixing in a script-mixing setting. [7] analyse the Hi-En Facebook posts and note that there is significant amount of Code mixing in the form of En in Hi matrix and vice versa. In a recent study, [25], authors present a large dataset of Code switched posts collected from multiple multilingual discussion forums on Reddit, and present a corpus comprising of Code mixing between En and 10 other languages and try to identify the map the reasons for Code mixing in written format as compared to spoken form.

The following chapters of this report are based on the text as a source of Code mix data, and within that, data sourced from online social networks occupies a primary role. The same also reflects in Section 5, where we list down the available data sources of Code mix sentences.

3.3 Language Identification

Language Identification of a text as one of the given languages is considered to be a solved task, using simple n-gram approaches, character level methods and stop words list, with the assumption that the text belongs to only one language [26]. But the same for mixed language text requires finer level analysis, breaking down the task as a token level language identification. LI accuracy is critical as it is often the first step in longer text processing pipelines, so errors made in LI will propagate and degrade the performance of later stages, and the nature of pre-processing hugely impacts the performance of Language Identification tool, especially for short text and noisy text from online social networks [27]. Language Identification on Code mix text is one of the most well-studied task on Code mix text, building tools suited for Code mix text. However, the performance measures for such tasks has remained in mid 90s [26]. Transliteration of words when writing words from native languages, and the spelling variations in such transliterations are a challenge for a language identification tool.

In annotating data for such task, the blurred boundaries between Code mixing and borrowing can result in noisy annotations. [28] give the example of token "glass" for English-Hindi language pair. Word level Code mixing can make this task all the more challenging. An example shown by [7] is "Computeron", which has a Hindi plural suffix "-on" over the English word "computer" is ambiguous - do we annotate it as Hindi or English word. Further, in most of such datasets, the language pair is known apriori. However, when analysing text from social networks, such a language prior can't be assumed, although we can use some heuristics to guess one, like the nature of query term used to collect data, geotags that particular text etc. Thus a robust language identification tool that has a wide coverage, i.e can disambiguate between multiple languages, spanning different language families (Dravidian, Indo-European in case of Indian context) is of crucial part of a text processing pipeline, and has to be one of the necessary steps of pre-processing of Code mix text. Number of languages between which a LID tool can disambiguate has to be carefully balanced, with the accuracy per language.

3.4 Transliteration, Lack of Standard Spellings and Need for Appropriate Pre-processing

Social media text is noisy in general. Even if we look at a social media post or comment written only in one language, we will come across the peculiarities of text in social media : use of hashtags, user mentions, emojis, multiple punctuation marks etc. While they bring in their own semantic and pragmatic value, nonetheless, they are challenging to any automatic text processing pipeline. Further social media text is more prone to contractions due to brevity constraints, usage of non-standard spellings and abbreviations ("gn", "OMG", "YOLO"), and spelling mistakes.

In addition to such quirks of social media text, the complexity in automatic processing of social media is compounded by Code mixed and mixed script posts. In social media, the dominant language is English with a growing share of regional languages. If a user were to post something in their native language they have two options : a) write in native script (Devanagari) or b) transliterate in roman script. In the case of transliterated text, as there are no standard spellings, users can transliterate the same word in any number of ways. Both of these ways of writing one's native language render the automatic processing of social media text all the more challenging.

One of the remedial steps to overcome this challenge would be to have a robust and strong normalisation block higher up in the text processing pipeline. A normalisation engine that is capable of dealing with the general quirks of social media text and also the nuances of mixed script text, transliterated text and noisy spelling variations.

3.5 Language Pairs - The sheet variety

Code mixing as a phenomena is prevalent in all multilingual communities. On Online Social networks, all the regional languages are likely to be present. According to the 2011 Census , 26% of the population of India is bilingual, while 7% is trilingual, placing India uniquely in terms of wide variety of Code

mixing language pairs.

[8] report that in the context of European languages, Code mix research has been primarily focused on Turkish-Dutch, Frisian-Dutch, Turkish-German and Ukranian-Russian with some initial attempts being made in parsing Russian-Komi text, while in the context of Indian Languages, Hindi-English is the most widely studied language pair for computational processing, with some recent work on Telugu-English, Tamil-English, Bengali-English and Gujarati-English.

Recently there have been few datasets published for English-Dravidian language pairs like English - Telugu, English-Tamil, English-Malayalam. However, any language pair other than En-Hi has much higher degree of resource constraints. Further, certain topics on online social networks can have posts from a wider diaspora of users, thus posts which needn't have only one kind of language mixing, but Code mixing and mixed scripts from multiple languages. This multilingual nature of text on social networks can have adverse impact on performance of any text processing pipelines.

This variety in the language mixes, and the associated resource availability concerns are an impediment in scaling the current automatic text processing pipelines to Code mix text and text from Online Social network in general.

4 Measures of Code Mixing

The linguistic studies aim to characterize the nature of Code mixing, are qualitative and focused on the contact between languages, and the factors influencing such Code mixing. However, as we scale to larger corpora sizes, to different languages, we need a metric(s) to measure the degree of Code mixing given a sentence or a corpus. Such metrics are useful in characterizing the multi lingual documents : Is the corpus altering between two languages in sentences or is there Code mixing in a single utterance/sentence? ; Is the Code mixing occurring only for lone lexical items and multi word expressions (MWE) or are the there instances of intra-word Code mixing? Understanding the nature of Code mixing can help us design the algorithms and tools for automatic processing of Code mixing text, and text from social media in general [29].

[29] list different measures of Code mixing, categorised into different kinds, along with their motivation.

1. Ratio

    - M-Index

      Developed from Gini Coeffecient, M-index is a word-count-based measure that quantifies the inequality of the distribution of language tags in a corpus of at least two languages. The M-index is calculated as follows, where k $>$ 1 is the total number of languages represented in the corpus, p_j is the total number of words in the language j over the total number of words in the corpus, and j ranges over the languages present in the corpus:

$$M - index = \frac{1 - \sum p_j^2}{(k-1).\sum p_j^2}$$

The index is bounded between 0 (monolingual corpus) and 1 (each language in the corpus is represented by an equal number of tokens).

- Language Entropy Language entropy of a corpus. The language entropy returns how many bits of information are needed to describe the distribution of language tags.

$$LE = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

- Probability of Switching (I-index)

  Integration-Index, a metric that describes the probability of switching within a text. Let us define any token in the corpus that is preceded by a token with a different language tag as a switch point. Then the I-index is a proportion of how many switch points exist relative to the number of language-dependent tokens in the corpus. In other words, it is the approximate probability that any given token in the corpus is a switch point. Given a corpus composed of tokens tagged by language l_i where j ranges from 1 to n, the size of the corpus, and i = j − 1, the I-index is calculated by the expression

$$I - Index = \frac{1}{n-1} \sum_{1 \leq i = j-1 \leq n-1} S(l_i, l_j$$

  where $S(l_i, l_j) = 1$ if $l_i \neq= l_j$ and 0 otherwise, and the factor of 1/(n − 1) reflects the fact that there are n − 1 possible switch sites in a corpus of size n. This index has utility for differentiating between corpora that are similarly multilingual but contain different patterns of switching behavior. For example, a parallel corpus would return an I-index very close to zero, whereas a corpus containing classic C-S would return values relatively farther away from zero.

2. Time-Course Measures : These measures go beyond the simple word counts and include the information about the temporal distribution of Code mixing across corpus.

- Burstiness : measures the manner and extent to which observed C-S behavior differs from a Poisson process (i.e., a process in which switching occurs at random). Briefly stated, it quantifies whether switching occurs in bursts or has a more periodic character.

  Let σ_τ denote the standard deviation of the language spans and $m\tau$ the mean of the language spans. Burstiness is calculated

$$Burstiness = \frac{\sigma_\tau / m_\tau - 1}{\sigma_\tau / m_\tau + 1} = \frac{\sigma_\tau - m_\tau}{\sigma_\tau + m_\tau}$$

  and is bounded within the interval $[-1, 1]$. Corpora with antibursty, periodic dispersions of switch points take on burstiness values closer to -1. By contrast, corpora with less predictable patterns of switching take on values closer to 1.

- Span Entropy : returns how many bits of information are needed to describe the distribution of the language spans. Let M denote the total number of states within the language span distribution, and l denote a specific span within that distribution where p_l represents the sample probability of a span of length l. The span entropy is then defined as

$$LE = -\sum_{l=1}^{M} p_l \log_2(p_l)$$

3. Memory : Although the burstiness and span entropy metrics take into account the time spacing between switch points, they cannot make claims about the time ordering of the language spans. It is possible for two corpora to have identical language span distributions – and thus the same Burstiness-index – that nonetheless appear very different due to how the switch points are ordered. Memory metric proposed by [30] quantifies the extent to which the length of language spans tend to be influenced by the length of spans preceding them.

   Let n_r be the number of language spans in the distribution and τ_i denote a specific language span in that distribution ordered by i. Let σ_1 and m_1 be the standard deviation and mean of all language spans but the last, where σ_2 and m_2 are the standard deviation and mean of all language spans but the first.

   Memory is calculated as

$$Memory = \frac{1}{n_r - 1}\sum_{i=1} n_r - 1\frac{(\tau_i - m_i)(\tau_(i+1) - m_2)}{\sigma_1\sigma_2}$$

   and is bounded within the interval [-1,1]. Memory values close to -1 describe the tendency for consecutive language spans to be negatively autocorrelated, differing substantially in length; that is, long spans of discourse are followed by short spans of discourse, and short spans are followed by long spans. Conversely, memory values closer to 1 describe the tendency for consecutive language spans to be positively autocorrelated, meaning similar in length.

The distribution-based measures, burstiness and span entropy, and the time series measure, memory, complement one another to describe the intermittency of switching behavior. In concert with the other metrics, which give a sense of the extent of language mixing, these measures relay a comprehensive signature of C-S for any language-tagged corpora.

[31] introduce a new metric called Code mixing Index (CMI), to evaluate and compare level of Code mixing at utterance level.

$$CMI = \begin{cases} 100 * [1 - \frac{max(w_i)}{n-u}], & \text{if } n > u \\ 0, & \text{if } n = u \end{cases}$$

where w_i is the words tagged with each language tag $\sum_1^N(w_i)$ is the sum over all N languages present in the utterance of their respective number of words, $maxw_i$ 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 (in our case that means tokens tagged as "universal", as abbreviations, and as named entities).

[32] note that CMI doesn't reflect the fraction of corpus's utterances contain Code switching and that CMI doesn't take into account the number of Code alteration points, and propose a modified version of CMI for utterance level. Moving upto corpus level, the authors also propose a corpus level Code mix measure which takes into account Code alteration between two utterances, and take inspiration from readability indices to propose a measure for corpus level Code mix measure.

For utterance level :

$$\begin{aligned} C_u(x) &= w_m f_m(x) + w_p f_p(x) \\ &= w_m \frac{N(x) - \max_{L_i \in \mathbb{L}}\{t_{L_i}\}(x)}{N(x)}.100 + w_p \frac{P(x)}{N(x)}.100 \end{aligned} \tag{1}$$

where x is the utterance, $L_i \in \mathbb{L}$ is the set of all languages. $P(x)$ is the number of Code alternation points, and w_m and w_p are weights ($w_m + w_p = 1$). Again, $C_u = 0$ for mono-lingual utterances (since in that case $\max\{t_{L_i}\}$= NandP $= 0$).

For Corpus level :

$$\begin{aligned} C_c &= \frac{\sum_{x=1}^{U} C_u(x) + w_p \delta(x)}{U} + w_s \frac{S}{U}.100 \\ &= \frac{100}{U}\left[\sum_{x=1}^{U} \left(w_m f_m(x) + w_p \left[f_p(x) + \delta(x)\right]\right) + w_s S\right] \end{aligned} \tag{2}$$

where w_m and w_p are weights ($w_m + w_p = 1$), where S is the number of utterances that contain Codeswitching ($0 \leq S \leq U$), w_s is the relative weight attached to the switching frequency, and $\delta(x)$ is 1 if language with maximum tokens for utterance x is not same as language with maximum tokens for utterance $x - 1$, otherwise 0.

However, it is worth noting that all the aforementioned metrics are computed only on the token wise language IDs. Weather or not they capture the syntactically complexity of the Code mix utterance needs to be established. A sentence where a single word is switched is not syntactically complicated as compared to word level morphological variation which can be attributed to a another language, or a sentence where a longer span is switched. Publicly available Code mix datasets (as mentioned in 5) haven't commented on the syntactical complexity of the samples in the datasets, to the best of our knowledge. This poses a crucial question for computational models for Code mix sentences - are there Code mix samples of particular syntactical category that are harder to process? This line of inquiry also opens up scope for insightful error analysis for the computational approaches to Code mixing.

In addition to the metrics mentioned in [1, 28, 32, 33] have attempted to identify shortcomings of these measures from following perspectives :

- Metric formulation : current formulation where only the language IDs are taken into account, can give higher metric for meaningless and/or unnatural Code mix sentences.
- Resource limitation and noisy human LID annotation : existing Code mix datasets have noisy, and monolingual sentences, and poor quality LID tool impact the usability of existing Code mix metrics.

To further analyse the usability of these metrics, authors also manually annotate Code mix sentences on two dimensions :

- **Degree of Code mixing** : score on the scale of 0 to 10 where 0 indicates monolingual sentences without any Code mixing, while a score of 10 indicates high degree of Code mixing.
- **Readability** : score on the scale of 0 to 10 where 0 indicates completely unreadable sentence based due to large number of spellings mistakes, lack of sentence structure and meaning, while a score of 10 indicates highly readable sentence with clear semantics and easy-to-read.

A similar human annotation approach is adopted by [34] to estimate the quality of generated Code mix sentences, by tasking the human annotators to give a score, between 1 and 5, along three dimensions: Syntactic, Semantic and Naturalness.

5 Data, Resources and Tasks

Code mix data, generally, is available across two sources: spoken data, social media data. Text from online social networks, even if it's not Code mixed, processing it has its own challenges : non-Canonical in their orthography, lexicon, syntax, thereby necessitating tools or computational pipelines that are able to automatically process such text.

5.1 Datasets and Tasks

In Table 1 we list down various datasets and tasks that have been proposed and released across Code mix setting of Indian languages. Additionally, [35] provide a comprehensive survey of research in computational processing of C-S text and speech and [5] present a list of datasets available for C-S research.

Name	**Language Mix**	**Source of dataset**	**Purpose of Dataset**
LINCE Benchmark [36]	hi-en, es-en, ne-en, MSA-Egyptian Arabic	Tweets, Facebook, Conversational	LID
			POS
			NER
			MT
GLUECoS Benchmark [37]	en-es, en-hi	Tweets, Facebook, Translated monolingual datasets	LID
			POS
			POS
			NER
			Sentiment Analysis
			NLI
			QA
Sentiment Analysis [38]	en-hi	Tweets	Sentiment Analysis
Semeval-2020 Sentiment Analysis [39]		Tweets	Sentiment Analysis
Machine Translation [40]	en-hi	Social Media	MT
Aggression Detection Shared Task [41]	en-hi	Facebook, Twitter	Aggression Detection
Hate Speech Detection [42]	en-hi	Tweets	Hate speech detection
Stance Detection [43]	en-hi	Tweets	Stance Detection
Stance Detection [44]	en-hi	Tweets	
Stance Detection [45]	en-ka	Facebook	
Sarcasm Detection [46]	en-hi	Tweets	Sarcasm Detection
Humor Detection [47]	en-hi	Tweets	Humor Detection
Code Mixed Goal Oriented Conversation Systems [48]	en-hi	Translated Monolingual Dataset	Conversational Datasets
	en-gu		
	en-ta		
	en-be		
Sentiment Analysis [49]	en-te	Tweets	Sentiment Analysis
ICON 2015-2016 Contest [50]	en-hi	Tweets, Facebook	POS, LID
	en-be		
	en-te		
Sentiment Analysis [51]	hi-en	Tweets	Sentiment Analysis
	bn-en		
FIRE 2013-16 Tasks [52]	en,hi,ba,gu,ml,ta,te	Tweets, Facebook, Gutenberg Project	Transliterated Search, Code Mix Cross Script QA, IR on Code mix hi-en tweets
Information Retrieval [53]	en-hi	Tweets	IR
FIRE 2020 Dravidian Code Mixed [54]	en-ta	YouTube Comments	Sentiment Analysis

	en-ml		
Offenseval Dravidian [55]	en-ta	YouTube Comments	Offensive Language Detection
	en-ma		
	en-ka		

Table 1: List of Code mix datasets, benchmarks, focused around English - Indian Language pair. Majority of these datasets are for en-hi language pair. Datasets have been released for various tasks, ranging from syntactic - LID, NER , Semantic - Sentiment, Hate/Offensive Sentence Detection, Discourse - Conversational. The proposed benchmarks are combination of syntactic and semantic tasks, providing a strong baseline to compare the efficacy of different representations, pipelines proposed by researchers.

5.2 Benchmarks

The computational methods and representations used for any downstream task should be able to generalize well across different datasets, language pairs and across different tasks. To test out efficacy of different methods and representations, having a common, standard benchmarks helpful. Recently two evaluation benchmarks,[37] [36], spanning different tasks and a few language pairs, thus enabling structured comparisons between different approaches. Figure 5 shows the various tasks, their language pairs, and the statistics of the corpus.

[37] released a benchmark named "GLUECoS", inspired by the GLUE benchmark, for language understanding evaluation for Code mixed languages. LID, PoS tagging, NER, Sentiment analysis, Question Answering, NLI tasks for English-Hindi language pair, and LID, PoS tagging , NER and Sentiment analysis for English - Spanish language pair.

English-Hindi				
Corpus	**Sent (Train)**	**Sent (Dev)**	**Sent (Test)**	**Sent (All)**
Fire LID (D)	2631	500	406	3537
UD POS (D)	1384	215	215	1814
FG POS (R)	2104	263	264	2631
IIITH NER (R)	2467	308	309	3084
SAIL Sentiment (R)	10080	1260	1261	12601
QA (R)	250	-	63	313
NLI (R)	1040	130	130	1300
English-Spanish				
Corpus	**Sent (Train)**	**Sent (Dev)**	**Sent (Test)**	**Sent (All)**
EMNLP 2014	10259	1140	3014	14413
Bangor POS	2192	274	274	2758
CALCS NER	27366	3420	3421	34208
Sentiment	1681	211	211	2103

Figure 5: Various Tasks and corpus statistics in GLUECoS Benchmark. (R) and (D) indicates Hindi written in Roman and Devanagari script, respectively

In addition to proposing the benchmark the authors also tested cross lingual word embedding

(MUSE [56] , BICVM embeddings trained on parallel data, BiSkip embeddings which uses parallel corpora and word alignments to learn cross-lingual embeddings, skip gram embeddings trained on synthetic Code mix data,), and mBERT (multilingual BERT pre-trained on monolingual corpora of 104 languages, and a modified mBERT fine tuned on Code mix data for Masked language model task.). Based on their experiments authors note that cross-lingual and mBERT models perform better for English - Spanish as compared to English - Hindi and attribute this to the English and Spanish are similar languages, and the lack of cannonical spellings for romanized Hindi, as we have already noted in the earlier section (Section 3.4). Authors also note that the modified nBERT model perform better for most tasks, and the accuracy vary for different tasks and are far from being good, except for LID. The performance measures as reported for sentiment analysis and NLI are in the range of 65-70%, demonstrating that there is a long way for result on Code mix datasets to be comparable with results of similar tasks on monolingual datasets.

[36], propose Linguistic Code switching Evaluation (LinCE) benchmark. LinCE benchmark has 4 language pairs as compared to the two language pairs proposed in GLUECoS benchmark, and covers 4 tasks, namely LID, NER, PoS tagging and Sentiment Analysis, while for some language pairs all the tasks are not present in the benchmark. Figure 6 shows overview of language pairs and tasks in LinCE benchmark, and 7 shows the dataset details for all the tasks.

Language Pair	LID	POS	NER	SA
Spanish-English	✓	✓	✓	✓
Hindi-English	✓	✓	✓	-
Nepali-English	✓	-	-	-
MS Arabic-Egyptian Arabic	✓	-	✓	-

Figure 6: Overview of language pairs and tasks in LinCE benchmark

Tasks	Corpus Authors	Languages	Training			Development			Test		
			CMI	Posts	Tokens	CMI	Posts	Tokens	CMI	Posts	Tokens
LID	Molina et al. (2016)	SPA-ENG	8.491	21,030	253,221	7.062	3,332	40,391	8.264	8,289	97,341
	Solorio et al. (2014)	NEP-ENG	20.322	8,451	122,952	17.079	1,332	19,273	19.754	3,228	46,559
	Mave et al. (2018)	HIN-ENG	10.222	4,823	95,224	10.122	744	15,446	9.930	1,854	36,052
	Molina et al. (2016)	MSA-EA	2.567	8,464	171,872	3.185	1,116	21,978	3.849	1,663	33,504
POS	Singh et al. (2018b)	HIN-ENG	21.449	1,030	22,993	15.293	160	3,476	18.910	299	6,541
	Soto and Hirschberg (2017)	SPA-ENG	24.191	27,893	217,068	24.040	4,298	33,345	24.282	10,720	82,656
NER	Aguilar et al. (2018)	SPA-ENG	5.567	33,611	404,428	4.398	10,085	122,656	5.867	23,527	281,579
	Singh et al. (2018a)	HIN-ENG	20.117	1,243	21,065	19.913	314	5,364	19.733	522	8,945
	Aguilar et al. (2018)	MSA-EA	–	10,103	204,296	–	1,122	22,742	–	1,110	21,414
SA	Patwa et al. (2020)	SPA-ENG	20.643	12,194	186,602	21.553	1,859	28,202	20.528	4,736	72,006

Figure 7: Corpus details for tasks in LINCE benchmark. Four language pairs - SPA-ENG, NEP-ENG, HIN-ENG, MSA-EA are part of the benchmark, with varying number of tasks - LID, PoS, NER, Sentiment Analysis.

Authors also provide baselines over these datasets using LSTM, ELMo and mBERT. Authors note that the pre-trained language models outperform simple BiLSTM models, with ELMo's performance being comparable with mBERT. Authors also note that NER and SA seem harder compared to LID and POS, with the NER and Sentiment Analysis have their performance measures low (NER Micro F1is in the range of 45-65 across language pairs) compared to those of LID and PoS(75-98% accuracy over language pairs). Similar observations was made in the context of GLUECoS benchmark.

However, it is worth noting that benchmarks only test certain capabilities of a representation or computational approach. When we compare the aforementioned benchmark with those of monolingual benchmarks (like GLUE, SuperGLUE), the number of tasks in the benchmark are fewer, thereby necessitating the expansion of such benchmarks - in terms of depth i.e number of tasks, and in terms of breadth - number of language pairs. Although the recently released benchmarks , which are amalgamation of tasks and datasets published by different researchers, the quality of data has to be critically examined.

While the benchmarks provide a structured way of comparing performance of computational approaches to Code mixing, only chasing incremental gain on such benchmarks needs to be avoided. [57] provides a critical view of benchmarking, and research in Code mixing needs to avoid similar pitfalls, and have a comprehensive view of data quality, be cognizant of shortcomings of any computational approach, diversity of language pairs being mixed and their societal impact, instead of pure number-chasing on a benchmark.

5.3 Computational Approaches

As listed in table 1, various tasks have been explored in Code mixing of Indian languages, across language pairs with en-hi being the most dominant mix among the available datasets. However, despite significant efforts, language technologies are not yet capable of processing C-S as seamlessly as monolingual data. We identify three main limitations of the current state of computational processing of C-S: data, evaluation and user-facing applications [8].

Application and performance of Deep Neural networks on Code mix tasks is constrained by the amount of data available for a particular language pair. Since C-S languages tend to be low resourced, building Deep Learning- based models is challenging due to the lack of large C-S datasets. Low-resource setting is idle for leveraging transfer learning and data available from massive monolingual corpora. Transformer based models like BERT [58] and plethora of models that have followed from Transformer-based architecture have been heavily applied in monolingual setting, and have outperformed traditional models on several benchmark tasks. Similar architectures [58, 59] have been proposed in multilingual setting as well, and have been tested for their zero-shot, few-shot learning. Naturally, these models have been tested on the Code mix tasks as well, and have outperformed task specific models [37, 60], but their performance is way worse than their performance on similar tasks in monolingual setting [8]. Since these models haven't been trained specifically on the scarce Code mix corpora, it is likely that performance of large multilingual LMs will improve on Code mix tasks. At the same time, this calls for deeper inquiry into what makes large multilingual LMs perform well on

Code mix tasks. A critical analysis on the data quality , combined with deeper look into inner working of the models, like [61], are crucial for improving performance on various Code mix tasks. Further usage of synthetic Code mix data has been explored in recent works on Code mix tasks - Sentence classification, Machine Translation - [37, 34, 62]. However, [37] noted that adaptive pre-training of models using real Code mix data performs better than model pretrained on synthetic Code mix data.

6 Gaps Identified

6.1 Efficient Code mix Data Collection, Pre-processing and Annotation Methods

When we aim to develop computational tools that are generalized and capable of processing Code mix data we need rich representations which need large corpora to train but Code mix data is very low resource. Code mixed is less likely to occur in traditional sources of large corpora - like news articles, wikipedia. And thus researchers have relied on speech corpora, text messages corpora, corpora from online social networks. As we have stated in Section 3 there are privacy concerns around collection of such data. As social network companies come under increasing pressure to curb the data leaks and privacy infringements, sourcing data from social networks is getting increasingly difficult. Thus we need more efficient data collection methods - for example : finding high yield Code mix terms from corpora by filtering the Code mix query terms, and have to look at alternate sources of data - for example : comments on newspaper articles, on popular video sharing platforms.

Additionally, as we build tools capable of processing social media text - such tools shouldn't be domain specific or dataset specific. To this end, we need methods that are capable of handling the transiency and the variety of textual data on social media. In such a setting, annotating data will have an impact on the time needed to develop a tool ergo the utility of the tool, and the need to repeated manual annotation of data is a huge obstacle. To overcome this we need methods that do not overly rely on quick annotation methods. [63] demonstrate that understanding tweets is dependent on hashtags as they enCode the affective and semantic content, and having a reliable hashtag segmentation tool can be used to create noisy training data. Further exploration of such methods and leveraging them to build tools that have quick turn around time for a specific task will have a huge impact on the usability of a toolkit for understanding Code mix data and also social media text in general.

6.2 Transfer Learning from Large Multilingual Models, Monolingual Corpora for Richer Representations for Code Mixing Tasks

As Code mix is low resource, the monolingual corpora which are available should be effectively leveraged to understand the Code mix data. [64] demonstrate the utility of monolingual resoucres in dependency parsing of Code mix sentences. In [37, 36], GLUECoS and LinCE benchmarks, the authors provide the baseline results using mBERT which is a large LM trained on monolingual corpora of

104 languages. The rationale of using the multilingual approach is that the a model like BERT, which has 110M parameters in its base version, when trained on multiple languages is good at zero-shot cross-lingual model transfer, in which task-specific annotations in one language are used to fine-tune the model for evaluation in another language. Similar usage of multilingual models in Code mix setting is demonstrated by the submissions in the Semeval 2019 shared task on Sentiment analysis on Code mix tweets [39], an by [65] for a semi supervised approach for generating Code mix sentences. Further, transfer learning can be a utilised when generalizing the Code mix models to other low resource Code mix language pairs like English - Telugu, English-kannada etc for which resources are very few.

Although in recent past, usage of models like XLM and mBERT has increased for Code mix settings, the usage of such tools has to be critically examined. [66, 67] has shown the sensitivity of BERT to noisy text (spelling variations, typos, contractions etc) and the associated degradation in performance. Further, [60] analyse the performance of mBERT on Code mix tasks and note that the sub par performance of mBERT for transliterated text. However, recently Google released its BERT model, named MuRIL, trained on monolingual corpora of Indian Languages along with their transliterated counterparts. It is worth comparing the performance of aforementioned large LMs on Code mix benchmarks and reporting the results. In our initial experiments with XLM, we have noted sub par performance of XLM on Code mix Masked language model task, and find that the language embeddings aren't enforced on the final prediction as much as we want. This clearly shows the need to adopt multilingual large LMs to Code mix tasks.

Bias of large LMs : [68, 69, 70] point out that the large LMs which are trained on large corpora from web, suffer from toxic and biased behaviour. These models have been shown to exhibit rasicts, sexist and toxic behaviour. The problem is further reinforced when the trianing corpora for such models is extracted from User Generated Content from web. Majority of data resources for Code mix research have been sourced form Online social networks which is likely to have hate speech, toxic language. In [37] use a modified mBERT fine tuned on Code mix corpora and demonstrate improved performance measures across different tasks. As researchers, we need to be mindful of the harmful effects of deploymnet of such models. Thus, characterising the harmfulness of such models, in the Code mix setting, has to be advised whenever a model is trained on the corpora extracted from online social network.

6.3 Lack of Standard Pipelines for Processing Code Mix Text for Applicaitons

While there has been various tasks and associated datasets for Code mix, they have been treated on an individual basis and the computational approaches were also suited for that task. But as the Code mix research community looks to solve the tasks higher up in the NLU ladder, we have seen sub par performance on such tasks. As an example, Although the research has shown that the crucial steps of LID and normalisation have a positive impact on the performance, such measures have been ignored while applying the recent multilingual large LM models. Currently, integrating all the different methods (LID, transliteration, normalisation, representations) for a single task have to be configured

and accessed from different resources and libraries, which is time consuming and in turn effect the quick prototyping. Integrating all the tasks together in a common pipeline/framework, and a library - a common place to access different resources (datasets, corpora), and collection of computational methods known to work well for Code mix data - that facilitates a integrated approach is really needed to further work in Code mix community

The end goal of a comprehensive toolkit / framework capable of automatically processing noisy Code mix text from online social networks. Such a toolkit will have multiple use cases, particularly in text classification tasks on data from social media. For instance, problem like open domain stance detection on online social networks. Sentiment analysis gives very limited view of the opinion held for/against a particular target. While Stance Detection, solves this problem, a Stance Detection toolkit that is capable of accurately detecting on zero-shot targets is an unsolved problem, particularly for noisy posts from online social networks. Such a tool's utility lies in the fact that topics of interests on social networks are very transient, and the need to having to repeatedly annotate target specific data is a huge obstacle. To quote a few more examples for Privacy and Security domain are detection of hateful and offensive content, and detection of phishing posts. To have a reusable framework that is tested on a subset of such problems would demonstrate the efficacy and usability.

7 Current Work Under Progress

Over the course of last 4 semesters, I have worked over various problems for processing text from social networks, trying to identify the methods that could be useful to stitch together a pipeline capable of processing noisy text. The on-going of my work are listed below:

7.1 Code Mix Data Collection

As mentioned in Section 3, one of the primary challenges in computational processing of Code mixing is availability of large, high-quality Code mix corpora. Researchers have used various heurestics to collect Code mix data for specific tasks (Sentimens analysis). There are two primary problems with heuristic based approaches used hitherto:

1. Usability of the methods listed above depend on the strength of the heuristic.
2. Further, given the fact that only a small proportion of the collected utterances are likely to be Code mix, the precision is low in these methods.

To address this gap, we are prototyping a pipeline for collecting Code mix corpora. Following are the essential components of this pipeline.

1. **Sentence level Classifier:** Filtering out Code mix sentences from a large corpus - this is a common problem for researchers creating a Code mix resource. Here, we formulate the problem

as a two step classification problem - First, classify weather a sentence is Code mix or not, if it is, then what is the language mix. The language mixes are en-IL pairs (en-hi, en-bn etc)

2. **High Code Mix yield Query term for Social Media APIs** : The primary research question in this work is : Are their certain query terms, which are mix of tokens from a language pair (ex: "I\en wish\en ki\hi.."), which will have higher proportion of Code mix utterances? If yes, how can we mine them from a large collection of Code mix sentences, while avoiding the bias these selective query terms bring in?

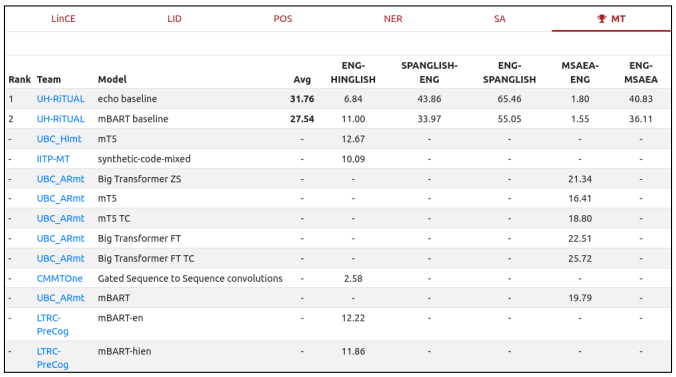
3. **Hashtag Segmentation** : Hashtags have become ubiquitous in across online social networks. Hashtags have a lot of semantic value. However, unsegmented nature of hashtags pose a challenge to leverage semantics enCoded in the hashtag. Ex. "MainBhiChowkidar", "ChowkidarChorHai". While the previous work on hashtag segmentation [71, 63] have proposed datasets and architectures which have performed well. However, their performance is sub-par for hashtags which have named entities, and specifically named entities contextual to Indian subcontinent. We are testing this hypothesis by annotating set of hashtags where the existing SOTA architectures fail, and formulating pipeline to overcome this challenge.

7.2 Quantitative and Qualitative Analysis of Code Mix Utterances

The Code mix measures listed in Section 4 consider only the token wise language tags of the utterances and propose measures which provide different perspective of Code mix pattern observed. However, these measures are lacking in capturing the syntactical complexity involved in the utterance. In this body of work, we are focusing on en-hi language pair, and collecting the publicly available datasets to have a large collection of Code mix utterances, apply a PoS tagger to the collected utterances, and analyze patterns in the corpus.

7.3 Publications

- **Shared Task on Code Mix Machine Translation**: We competed in shared task on Code Mix translation hosted at CALCS 2021, and our system scored 2nd on the en-hi language pair leaderboard [72].

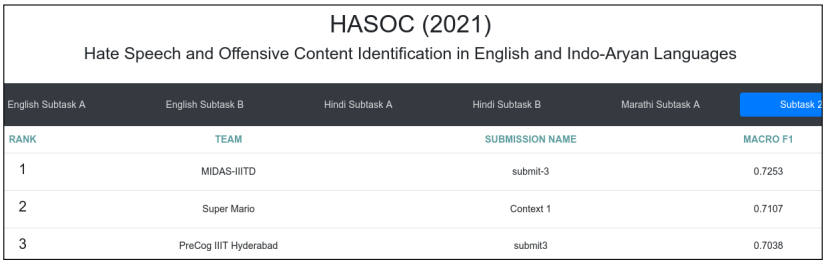


LinCE | LID | POS | NER | SA | MT

Rank	Team	Model	Avg	ENG-HINGLISH	SPANGLISH-ENG	ENG-SPANGLISH	MSAEA-ENG	ENG-MSAEA
1	UH-RiTUAL	echo baseline	**31.76**	6.84	43.86	65.46	1.80	40.83
2	UH-RiTUAL	mBART baseline	**27.54**	11.00	33.97	55.05	1.55	36.11
-	UBC_Hlmt	mT5	-	12.67	-	-	-	-
-	IITP-MT	synthetic-code-mixed	-	10.09	-	-	-	-
-	UBC_ARmt	Big Transformer ZS	-	-	-	-	21.34	-
-	UBC_ARmt	mT5	-	-	-	-	16.41	-
-	UBC_ARmt	mT5 TC	-	-	-	-	18.80	-
-	UBC_ARmt	Big Transformer FT	-	-	-	-	22.51	-
-	UBC_ARmt	Big Transformer FT TC	-	-	-	-	25.72	-
-	CMMTOne	Gated Sequence to Sequence convolutions	-	2.58	-	-	-	-
-	UBC_ARmt	mBART	-	-	-	-	19.79	-
-	LTRC-PreCog	mBART-en	-	12.22	-	-	-	-
-	LTRC-PreCog	mBART-hien	-	11.86	-	-	-	-

Figure 8: Screenshot from LINCE Benchmark Leaderborard for Code Mix Machine Translation task. We competed for en-hi language pair, and our system scored 2nd best for en-hi pair.

- **Shared Task on Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages HASOC 21**: We competed in shared task on Code Mix Hate Speech and Offensive Content Identification HASOC 2021, and our system scored 3rd on the en-hi language pair leaderboard. The paper is under review currently.



HASOC (2021)

Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages

English Subtask A | English Subtask B | Hindi Subtask A | Hindi Subtask B | Marathi Subtask A | Subtask 2

RANK	TEAM	SUBMISSION NAME	MACRO F1
1	MIDAS-IIITD	submit-3	0.7253
2	Super Mario	Context 1	0.7107
3	PreCog IIIT Hyderabad	submit3	0.7038

Figure 9: Screenshot from Leaderborard for en-hi Code Mix Hate Speech classification. Our system ranked 3rd in the task.

Current work is likely to be in Draft Stage by November 2021.

8 Future Work

As the current work matures, there will be additional aspects integrated into them, and in addition to that, following are some of the planned future work.

8.1 Bias Evaluation of Different Models

[73, 74] provide methods for quantifying bias for different contextual word embeddings. Since these works are focused around English, the methods have to be improvised and adapted for multilingual and Code mixed setting involving Indian languages.

Data Collection, Pre-Processing	Transfer Learning, Representation for CM	End-to-end pipeline	Other
Sentence Level CM Classifier	Analysis of large multlingual LMs for Code mixing	Machine Translation, Generation	Bias of Models trained on Code mix Data
Syntactic Analysis of Code Mix data	Modifying XLM-R for richer CM Representations	Benchmarks of CM	

Table 2: In this table, we list all the current work, and future work and map it to the gaps identified in Section 6. The cells marked in Green are currently in progress, cells marked in purple indicate that the initial work didn't yield favourable results and needs reformulation, cells marked in yellow indicate that they are part of future work and needs to be explored. It should be noted that, works categorized under different heads will contribute in the End-to-end pipeline. Rationale for specifically mentioning 'End-to-end Pipeline' as a separate category is to mention the CM tasks we plan to explore.

8.2 Machine Translation and Generation

Training a SOTA MT model for faithful translation between Code mix and constituent monolingual languages, is at a very nascent stage. Primarily because of unavailability of parallel corpora at the scale needed to train large models like Transformers. Unsupervised MT, Transfer learning using multilingual models were the preferred methods in recent publications on the problem. In addition to our recent shared task experience (Section 7.3) we wish to explore the bi-directional Machine translation between Code mix - monolingual parallel corpora. Given the low resource nature of Code mix, for creating large corpora of Code mix sentences across rarer language pairs, MT using unsupervised methods like the one proposed in [34]. Further, development of neural architectures that are capable of generating Code mix, and controllable for different metrics and quality of Code mix.

8.3 Comparison of Different Multilingual Representations

With the GLUECoS, LinCE benchmak, we can now evaluate and compare the performance of different multilingual models for different Code mix tasks. In recent work on the shared tasks, and these benchmarks, large multilingual LMs like mBERT, MuRIL, XLM have been experimented with, although the results indicate that the problems remain unsolved. In addition to the evaluation of these models, we can tweak the models for their pre-training objectives, and/or modify their loss functions to enhance performance of such multilingual models in a Code mix setting.

9 Limitations

- Although code mixing is also prevalent in speech data, in this study we have focused only text as a source of code mix data, and the observations/comments made regarding computational approaches to code mixing are made in the context of automatic text processing pipelines.
- There are many lenses through which one can analyse the phenomena of code mixing : socio- and psycho-linguistic, grammatical theories etc. In this study we focus on automatic text processing capabilities of text processing pipelines, and how to enhance them.
- We have noted in previous sections that code mixing is low resource. While we've listed data and resources for Indian Languages code mix pairs, the disparities, in terms of resource richness, within Indian Language pairs is quite stark. Researchers, so far, have focused on en-hi pair, more than any other and that poses a limitation on proposing and testing computational approaches that can be generalized to other code mix language pairs.

10 Selected Papers

Sno	Selected Paper	Author(s)	Rationale for selection
1	Bilingual speech : a typology of Code mixing. (Chapters 1,2) [2]	Pieter Muysken	Formulates categorisation of constraint based grammatical theories for Code mixing
2	Social and Psychological Factors in Language Mixing [3]	W. Ritchie, T. Bhatia	Provides the social and psychological factors underpinning multilinguals propensity to Code mix.
3	Challenges of computational processing of Code switching [26]	Çetinoğlu et al.	This paper lists down various challenge's for computational pipelines capable of processing Code mix data
4	A Survey of Code switching: Linguistic and Social Perspectives for Language Technologies [8]	Doğruöz et al	This is a survey of Code mixing from a linguistic and social perspectives which can dictate computational approaches to Code mixing, also listing challenges for computational approaches to Code mixing .
5	A Survey of Code switched Speech and Language Processing [35]	Sitaram et al	This paper provides a thorough survey of tasks proposed in Code mix setting, across language pairs, and presents brief summary of computational methods used for those tasks.
6	Metrics for modeling Code switching across corpora [29]	Guzmán et al	List and categorizes metrics used to assess complexity of Code mixing, and provides motivation behind formulation of those metrics
7	Comparing the level of Code switching in corpora. [32]	Gambäck et al	Proposes "Code Mixing Index (CMI)", a metric to assess Code mix complexity, and its variation for sentence level and corpus level metric
8	Challenges and limitations with the metrics measuring the complexity of Code mixed text. [33]	Srivastava et al	This paper identifies limitations of Code mixing measures, and provides a methodology for human annotation of Code mix complexity
9	LinCE: A Centralized Benchmark for Linguistic Code switching Evaluation. [36]	Aguilar et al	This paper provides a benchmark for Code mixing - spanning 4 language pairs and 5 tasks, to assess efficacy of computational pipeline and its generalizability

Sno	Selected Paper	Author(s)	Rationale for selection
10	GLUECoS: An evaluation benchmark for Code switched NLP [37]	Khanuja et al	Similar to LinCE benchmark, this paper proposes GLUECoS benchmark comprising of two language pairs and 7 tasks. Together these two benchmarks provide a framework for testing Code mixing pipelines and representations
11	BERTologiCoMix: How does Code mixing interact with multilingual BERT? [61]	Santy et al	Tries to explain why and what works well while using mBERT on Code mixing tasks/corpora.
12	From Machine Translation to Code Switching: Generating High-Quality Code Switched Text [34]	Tarunesh et al	Provides a unsupervised framework for generating synthetic Code mix sentences in Hindi frame.

References

[1] John J. Gumperz. *Conversational code switching*, page 59–99. Studies in Interactional Sociolinguistics. Cambridge University Press, 1982.

[2] Pieter Muysken. *Bilingual speech : a typology of code-mixing*. Cambridge University Press, Cambridge, UK New York, 2000.

[3] W. Ritchie and T. Bhatia. Social and psychological factors in language mixing. 2012.

[4] M. Gullberg, P. Indefrey, and P. Muysken. Research techniques for the study of code-switching. 2009.

[5] N. Jose, B. R. Chakravarthi, S. Suryawanshi, E. Sherly, and J. P. McCrae. A survey of current datasets for code-switching research. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 136–141, 2020.

[6] Utsab Barman. *Automatic Processing of Code-mixed Social Media Content*. Dublin City University, 2019.

[7] Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. "I am borrowing ya mixing ?" an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126, Doha, Qatar, October 2014. Association for Computational Linguistics.

[8] A. Seza Doğruöz, Sunayana Sitaram, Barbara E. Bullock, and Almeida Jacqueline Toribio. A survey of code-switching: Linguistic and social perspectives for language technologies. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1654–1666, Online, August 2021. Association for Computational Linguistics.

[9] Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 176–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[10] Wang Ling, Luís Marujo, Chris Dyer, Alan W. Black, and Isabel Trancoso. Crowdsourcing high-quality parallel data extraction from Twitter. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 426–436, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

[11] Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. Estimating code-switching on Twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1971–1982, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[12] E. Kim. Reasons and motivations for code-mixing and code-switching. 2006.

[13] Rafiya Begum, Kalika Bali, Monojit Choudhury, Koustav Rudra, and Niloy Ganguly. Functions of code-switching in tweets: An annotation framework and some initial experiments. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1644–1650, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[14] Sarah Thomason. Contact explanations in linguistics. 2020.

[15] Jasabanta Patro, Bidisha Samanta, Saurabh Singh, Abhipsa Basu, Prithwish Mukherjee, Monojit Choudhury, and Animesh Mukherjee. All that is English may be Hindi: Enhancing language identification through automatic ranking of the likeliness of word borrowing in social media. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2264–2274, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[16] J. Macswan. Code switching and grammatical theory. 2008.

[17] Carol Myers-Scotton and Janice Jake. A universal model of code-switching and bilingual language processing and production. *The Cambridge Handbook of Linguistic Code-switching*, pages 336–357, 01 2009.

[18] Mohd Sanad Zaki Rizvi, Anirudh Srinivasan, Tanuja Ganu, Monojit Choudhury, and Sunayana Sitaram. GCM: A toolkit for generating synthetic code-mixed text. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 205–211, Online, April 2021. Association for Computational Linguistics.

[19] Aravind K. Joshi. Processing of sentences with intra-sentential code-switching. In *Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics*, 1982.

[20] Itesh Sachdev, Howard Giles, and Anne Pauwels. *Accommodating Multilinguality*, chapter 16, pages 391–416. John Wiley Sons, Ltd.

[21] Suraj Maharjan, Elizabeth Blair, Steven Bethard, and Thamar Solorio. Developing language-tagged corpora for code-switching tweets. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 72–84, Denver, Colorado, USA, June 2015. Association for Computational Linguistics.

[22] John C. Paolillo. Language choice on soc.culture.punjab. 1996.

[23] John C. Paolillo. "conversational" codeswitching on usenet and internet relay chat. 2011.

[24] Abhishek Srivastava, Kalika Bali, and Monojit Choudhury. Understanding script-mixing: A case study of Hindi-English bilingual Twitter users. In *Proceedings of the The 4th Workshop on Computational Approaches to Code Switching*, pages 36–44, Marseille, France, May 2020. European Language Resources Association.

[25] Ella Rabinovich, Masih Sultani, and Suzanne Stevenson. CodeSwitch-Reddit: Exploration of written multilingual discourse in online discussion forums. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4776–4786, Hong Kong, China, November 2019. Association for Computational Linguistics.

[26] Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. Challenges of computational processing of code-switching. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 1–11, Austin, Texas, November 2016. Association for Computational Linguistics.

[27] Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. Automatic language identification in texts: A survey. *J. Artif. Int. Res.*, 65(1):675–682, May 2019.

[28] Amitava Das and Björn Gambäck. Code-mixing in social media text. the last language identification frontier? *Trait. Autom. des Langues*, 54:41–64, 2013.

[29] Gualberto A Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E Bullock, and Almeida Jacqueline Toribio. Metrics for modeling code-switching across corpora. In *INTERSPEECH*, pages 67–71, 2017.

[30] K.-I. Goh and A.-L. Barabási. Burstiness and memory in complex systems. *EPL (Europhysics Letters)*, 81(4):48002, jan 2008.

[31] Amitava Das and Björn Gambäck. Identifying languages at the word level in code-mixed Indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387, Goa, India, December 2014. NLP Association of India.

[32] Björn Gambäck and Amitava Das. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1850–1855, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[33] Vivek Srivastava and Mayank Singh. Challenges and limitations with the metrics measuring the complexity of code-mixed text. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 6–14, Online, June 2021. Association for Computational Linguistics.

[34] Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. From machine translation to code-switching: Generating high-quality code-switched text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3154–3169, Online, August 2021. Association for Computational Linguistics.

[35] Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W Black. A survey of code-switched speech and language processing, 2020.

[36] Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. LinCE: A Centralized Benchmark for Linguistic Code-switching Evaluation. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France, May 2020. European Language Resources Association.

[37] Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online, July 2020. Association for Computational Linguistics.

[38] Ameya Prabhu, Aditya Joshi, Manish Shrivastava, and Vasudeva Varma. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text, 2016.

[39] Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 774–790, Barcelona (online), December 2020. International Committee for Computational Linguistics.

[40] Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. Enabling code-mixed translation: Parallel corpus creation and MT augmentation approach. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 131–140, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[41] Ritesh Kumar, Aishwarya N. Reganti, Akshit Bhatia, and Tushar Maheshwari. Aggression-annotated corpus of Hindi-English code-mixed data. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[42] Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. A dataset of Hindi-English code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 36–41, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics.

[43] Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. An english-hindi code-mixed corpus: Stance annotation and baseline system, 2018.

[44] Sushmitha Reddy Sane, Suraj Tripathi, Koushik Reddy Sane, and Radhika Mamidi. Stance detection in code-mixed Hindi-English social media data using multi-task learning. In *Proceedings of the Tenth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 1–5, Minneapolis, USA, June 2019. Association for Computational Linguistics.

[45] V. Srinidhi Skanda, M. Anand Kumar, and K.P. Soman. Detecting stance in kannada social media code-mixed text using sentence embedding. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 964–969, 2017.

[46] Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. A corpus of english-hindi code-mixed tweets for sarcasm detection, 2018.

[47] Ankush Khandelwal, Sahil Swami, Syed S. Akhtar, and Manish Shrivastava. Humor detection in english-hindi code-mixed social media content : Corpus and baseline system, 2018.

[48] Suman Banerjee, Nikita Moghe, Siddhartha Arora, and Mitesh M. Khapra. A dataset for building code-mixed goal oriented conversation systems, 2018.

[49] S Padmaja, Sasidhar Bandu, and S Sameen Fatima. Text processing of telugu–english code mixed languages. In *Advances in Decision Sciences, Image Processing, Security and Computer Vision*, pages 147–155. Springer, 2020.

[50] Amitava Das. Tool contest on pos tagging for code-mixed indian social media (facebook, twitter, and whatsapp) text @ icon 2016.

[51] Braja Gopal Patra, Dipankar Das, and Amitava Das. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task @icon-2017. *CoRR*, abs/1803.06745, 2018.

[52] Somnath Banerjee, M. Choudhury, K. Chakma, S. Naskar, Amitava Das, Sivaji Bandyopadhyay, and P. Rosso. Msir@fire: A comprehensive report from 2013 to 2016. *SN Comput. Sci.*, 1:55, 2020.

[53] K. Chakma and Amitava Das. Cmir: A corpus for evaluation of code mixed information retrieval of hindi-english tweets. *Computación y Sistemas*, 20:425–434, 2016.

[54] Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Shardul Suryawanshi, Navya Jose, Elizabeth Sherly, and John P. McCrae. Overview of the track on sentiment analysis for dravidian languages in code-mixed text. In *Forum for Information Retrieval Evaluation*, FIRE 2020, page 21–24, New York, NY, USA, 2020. Association for Computing Machinery.

[55] Bharathi Raja Chakravarthi, Ruba Priyadharshini, Navya Jose, Anand Kumar M, Thomas Mandl, Prasanna Kumar Kumaresan, Rahul Ponnusamy, Hariharan R L, John P. McCrae, and Elizabeth Sherly. Findings of the shared task on offensive language identification in Tamil, Malayalam, and Kannada. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 133–145, Kyiv, April 2021. Association for Computational Linguistics.

[56] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.

[57] Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online, July 2020. Association for Computational Linguistics.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[59] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics.

[60] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics.

[61] Sebastin Santy, Anirudh Srinivasan, and Monojit Choudhury. BERTologiCoMix: How does code-mixing interact with multilingual BERT? In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 111–121, Kyiv, Ukraine, April 2021. Association for Computational Linguistics.

[62] Abhirut Gupta, Aditya Vavre, and Sunita Sarawagi. Training data augmentation for code-mixed translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5760–5766, Online, June 2021. Association for Computational Linguistics.

[63] Arda Çelebi and Arzucan Özgür. Segmenting hashtags using automatically created training data. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2981–2985, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[64] Irshad Bhat, Riyaz A. Bhat, Manish Shrivastava, and Dipti Sharma. Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 324–330, Valencia, Spain, April 2017. Association for Computational Linguistics.

[65] Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2267–2280, Online, November 2020. Association for Computational Linguistics.

[66] Ankit Kumar, Piyush Makhija, and Anuj Gupta. Noisy text data: Achilles' heel of BERT. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 16–21, Online, November 2020. Association for Computational Linguistics.

[67] Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy, July 2019. Association for Computational Linguistics.

[68] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3407–3412, Hong Kong, China, November 2019. Association for Computational Linguistics.

[69] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural*

Language Processing (EMNLP-IJCNLP), pages 2153–2162, Hong Kong, China, November 2019. Association for Computational Linguistics.

[70] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online, November 2020. Association for Computational Linguistics.

[71] Mounica Maddela, W. Xu, and Daniel Preotiuc-Pietro. Multi-task pairwise neural ranking for hashtag segmentation. In *ACL*, 2019.

[72] Devansh Gautam, Prashant Kodali, Kshitij Gupta, Anmol Goel, Manish Shrivastava, and Ponnurangam Kumaraguru. CoMeT: Towards code-mixed translation using parallel monolingual sentences. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 47–55, Online, June 2021. Association for Computational Linguistics.

[73] Keita Kurita, Nidhi Vyas, Ayush Pareek, Alan W Black, and Yulia Tsvetkov. Measuring bias in contextualized word representations. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 166–172, Florence, Italy, August 2019. Association for Computational Linguistics.

[74] Moin Nadeem, Anna Bethke, and Siva Reddy. Stereoset: Measuring stereotypical bias in pretrained language models. *ArXiv*, abs/2004.09456, 2020.