MIPE: A Metric Independent Pipeline for Effective Code-Mixed NLG Evaluation

Ayush Garg*, **Sammed S Kagi***
IIT Gandhinagar, India
{ayush.g,sammed.shantinath}@alumni.iitgn.ac.in

Vivek Srivastava
TCS Research, India
srivastava.vivek2@tcs.com

Mayank Singh
IIT Gandhinagar, India
singh.mayank@iitgn.ac.in

Abstract

Code-mixing is a phenomenon of mixing words and phrases from two or more languages in a single utterance of speech and text. Due to the high linguistic diversity, code-mixing presents several challenges in evaluating standard natural language generation (NLG) tasks. Various widely popular metrics perform poorly with the code-mixed NLG tasks. To address this challenge, we present a metric independent evaluation pipeline *MIPE* that significantly improves the correlation between evaluation metrics and human judgments on the generated code-mixed text. As a use case, we demonstrate the performance of *MIPE* on the machine-generated Hinglish (code-mixing of Hindi and English languages) sentences from the *HinGE* corpus. We can extend the proposed evaluation strategy to other code-mixed language pairs, NLG tasks, and evaluation metrics with minimal to no effort.

1 Introduction

Code-mixing (hereafter *'CM'*) is a commonly observed communication pattern for a multilingual speaker to mix words and phrases from multiple languages. CM is widespread across various language pairs across the globe, such as Spanish-English (Spanglish) and Hindi-English (Hinglish). Various studies (Baldauf, 2004) have predicted the high growth in the number of CM speakers, which would surpass the number of native speakers in various globally popular languages (e.g., English).

With the advent of social-media platforms (e.g., Twitter, Facebook, etc.), we observe a manifold increase in the CM communication by multilingual speakers. This leads to a large scale availability of CM data for various NLP tasks. Recently, we witness magnitude of work to address various CM NLP tasks such as language identification (Shekhar et al., 2020; Singh et al., 2018a; Ramanarayanan et al., 2019; Barman et al., 2014; Gundapu and Mamidi, 2018), POS tagging (Singh et al., 2018b; Vyas et al., 2014; Pratapa et al., 2018), named entity recognition (Singh et al., 2018a; Priyadharshini et al., 2020; Winata et al., 2019), word normalisation (Singh et al., 2018c; Parikh and Solorio, 2021), CM metrics (Guzmán et al., 2017; Srivastava and Singh, 2021a), sentiment analysis (Patwa et al., 2020; Joshi et al., 2016), stance detection (Utsav et al., 2020; Sane et al., 2019), natural language inference (Khanuja et al., 2020), machine translation (Srivastava and Singh, 2020; Dhar et al., 2018), and question-answering (Chandu et al., 2019; Thara et al., 2020).

We observe a growing interest in the computational linguistic community to study the CM NLG tasks. Recently, various resources and systems have been proposed that explore different dimensions of the CM NLG (Yang et al., 2020; Gautam et al., 2021; Gupta et al., 2021; Rizvi et al., 2021; Gupta et al., 2020; Jawahar et al., 2021). Evaluation of the CM NLG tasks is challenging due to the high linguistic diversity and lack of standardization. To address this challenge, Srivastava and Singh (2021b) has proposed *HinGE* corpus for the Hinglish CM text generation and evaluation (see Section 2 for details). *HinGE* corpus demonstrates the inefficacy of various widely popular metrics on the CM dataset.

In this paper, we choose five evaluation metrics (see Section 3 for details) as discussed in (Srivastava and Singh, 2021b) to demonstrate the efficacy of *MIPE*. Our proposed metric independent pipeline (*MIPE*) augments these metrics and addresses four major linguistic bottlenecks: (i) spelling variations, (ii) language switching, (iii) missing words, and (iv) the limited number of reference sentences associated with the CM NLG systems. The main contributions are:

- We identify four major reasons for the poor

* equal contribution

quality performance of various widely popular evaluation metrics for the code-mixed NLG evaluation.

- We propose a metric independent evaluation pipeline *MIPE* that addresses the identified bottlenecks in CM NLG evaluation. Furthermore, we show its efficacy in generating highly correlated metric scores against human scores.

The rest of the paper is organized as follows. In Section 2, we discuss the dataset for the CM NLG evaluation task. In Section 3, we present the *MIPE* pipeline addressing the four major bottlenecks for effective CM NLG evaluation. We discuss the results in Section 4. In Section 5, we discuss the current state and future direction. We conclude the discussion in Section 6.

2 Dataset

Recently, we observe various works to address the underlying challenges with the CM NLG. Numerous resources and systems have been proposed recently to advance the field. In our experiments, we use the *HinGE* corpus proposed in (Srivastava and Singh, 2021b). The *HinGE* corpus contains 1,976 English-Hindi parallel sentences from the IIT-B parallel corpus (Kunchukuttan et al., 2018). Corresponding to each of the English-Hindi parallel sentences, *HinGE* has two variants of CM Hinglish sentences:

- Human-generated Hinglish sentences: (Srivastava and Singh, 2021b) have employed eight human annotators to generate the Hinglish sentences. Each parallel sentence pair is annotated by a single human annotator. Human annotators have generated at least two Hinglish sentences corresponding to each parallel sentence pair. On average, 2.5 Hinglish sentences are generated for each parallel sentence pair.

- Machine-generated Hinglish sentences: Srivastava and Singh (2021b) proposes two rule-based algorithms to generate the CM sentences. They leverage the matrix-frame theory to generate the Hinglish sentences where Hindi is the matrix language and English tokens are embedded. The proposed algorithms differ significantly at the level of granularity (i.e., word and phrase). We will use the acronyms WAC (word-aligned code-mixing) and PAC (phrase-aligned code-mixing) for the two algorithm variants in the rest of the paper.

ENGLISH: is another human being saying, "Do you understand this?"
HINDI: कोई दूसरा मनुष्य ये कहे, "क्या आपको समझ आया?"
HUMAN-GENERATED 1: is another human being saying, "kya aapko samaj aaya?"
HUMAN-GENERATED 2: koi dusra human being yeh kahe, "Do you understand this?"

WAC GENERATED: koee doosra human ye kahe, kya aapko samajh aaya
RATING 1: 9
RATING 2: 8

PAC GENERATED: koee doosra manushy ye kahe, kya aapko samajh understand
RATING 1: 7
RATING 2: 8

Figure 1: Example of the CM sentences generated by the annotator along with WAC and PAC generated CM sentence from the parallel English-Hindi sentence pair. Two human annotators rate the machine-generated sentence on a scale of 1–10.

In addition to the machine-generated Hinglish sentences, *HinGE* has a human rating corresponding to each generated sentence. The human rating varies between 1–10, indicating low to high generation quality. Two human annotators have rated each of the machine-generated CM sentences. Figure 1 shows the example CM sentences generated by humans and two rule-based algorithms along with the rating to the machine-generated CM sentences. Figure 2 shows the distribution of the human ratings to the machine-generated Hinglish sentences. WAC-generated sentences receive a relatively high rating (> 6) as compared to PAC. In addition, WAC showed a low degree of human disagreements than PAC.

Efficacy of NLG Evaluation Metrics: Srivastava and Singh (2021b) present a study demonstrating the inefficacy of five widely popular NLG evaluation metric on the *HinGE* corpus. The five metrics are: (i) Bilingual Evaluation Understudy Score (**BLEU**, Papineni et al. (2002)), (ii) **NIST** (Doddington, 2002), (iii) BERTScore (**BS**, Zhang et al. (2019)), (iv) Word Error Rate (**WER**, Levenshtein (1966)), and (v) Translator Error Rate (**TER**, Snover et al. (2006)). Higher BLEU, NIST, or BS values and lower WER or

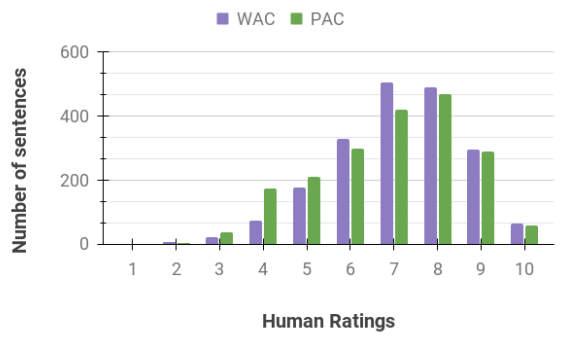

Figure 2: Distribution of human ratings on the generated Hinglish sentences using WAC and PAC. The figure is taken from Srivastava and Singh (2021b).

TER values represent better generation performance. Tables 1 and 2 show the comparison of five metric scores against the human ratings against WAC and PAC (see scores present in columns with heading *'Without MIPE'*). In addition, (Srivastava and Singh, 2021b) present a correlation study between the human ratings and the metric scores. For this purpose they divide the ratings into three buckets:

- Bucket 1: Human rating between 2–10.
- Bucket 2: Human rating between 2–5.
- Bucket 3: Human rating between 6–10.

Table 4 shows the correlation between the human ratings and the metric scores for WAC and PAC (see scores present in columns with heading *'Without MIPE'*). The correlation scores show a scope to build systems that shows a high correlation with human judgment.

3 MIPE

As discussed in the previous section, the widely used evaluation metrics fail to capture the linguistic diversity of the CM data. Based on the empirical observation on the 10 datasets used in (Srivastava and Singh, 2021a), we identify four major reasons for the failure of NLG evaluation metrics on the CM data. We propose a metric independent evaluation pipeline *MIPE*, for effective evaluation. Using *MIPE*, we first reduce the spelling variations (see Section 3.1) and the language switching (see Section 3.2) in the candidate Hinglish sentence. Next, we introduce a penalty (see Section 3.3) on the evaluation score based on the degree of importance of the missing words in the candidate Hinglish sentence. Finally, we address the challenge of a limited number of reference sentences (see Section 3.4) by segmenting the candidate and the reference sentences into phrases and leveraging the paraphrasing capability. Figure 3 shows the architecture of the proposed evaluation pipeline.

3.1 Spelling variations

The first challenge to effective evaluation is the non-standard spellings of the code-mixed words. E.g., words *kanekt*, *connect*, and *connekt* conveys the same meaning in a Hinglish sentence. Due to a lack of writing standards for the code-mixed languages, the speakers often use their phonetic understanding of the source languages to write the CM sentences. Hence, in most spelling variations, the addition, omission, and substitution of letters indicate that the phonetics remains almost the same. Specifically, we observe three major reasons for spelling variations,

- $R1$: character repetition
- $R2$: replacement with similar-sounding character
- $R3$: vowel omission

To address these problems, we normalize words such that similar-sounding words are grouped. We leverage the concept of Phonetic Dissimilarity (PDS, Toutanova and Moore (2002)) to address the spelling variations in the CM language. Our proposed PDS algorithm is a variant of the popular dynamic programming-based edit distance algorithm. Similar to edit distance, PDS quantifies the dissimilarity between two strings by counting the minimum number of edit operations (addition, deletion, and substitution) required to transform one string into the other. In PDS, we assign different costs to each edit operation based on the phonetic characteristics of the corresponding characters of the two words and the edit operation under consideration. To access the phonetic characteristics, we use a corpus of all possible pronunciations of the English alphabets[1]. Algorithm 1describes PDS between a word w_1 (in candidate CM sentence) and w_2 (in reference CM sentences). To address $R1$, we remove repeating characters from both words. By default, we keep addition and

[1] https://www.speakmethod.com/alphabet_sounds.html

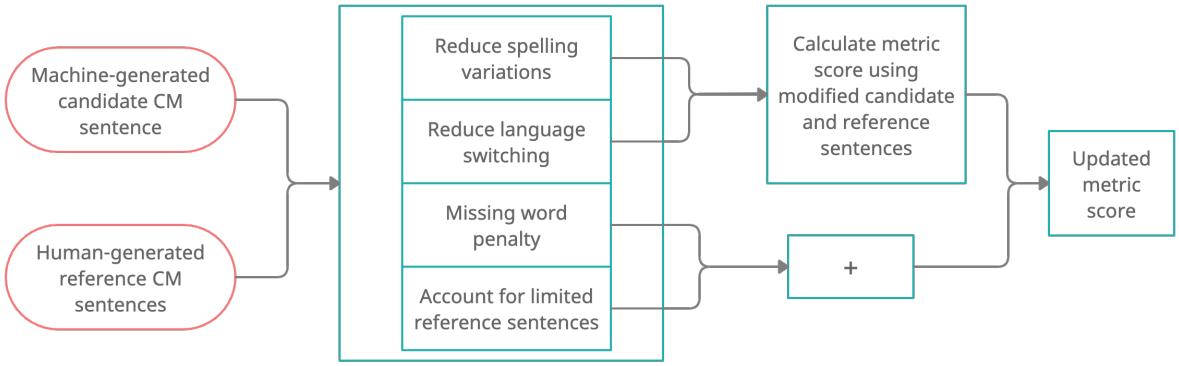

Figure 3: The architecture of the proposed CM NLG evaluation system *MIPE*. Machine-generated candidate CM sentences are generated by two rule-based algorithms (WAC and PAC). We reduce the spelling variation and language switching for both the candidate and reference sentences based on phonetics. A penalty is applied to the words in the candidate sentence which are not present in any of the reference sentences. We account for limited reference sentences by chunking the candidate and reference sentences into trigram phrases. The words in the candidate trigram phrases are assigned a score based on their presence in the reference phrases. The candidate phrase score is used to account for the limited reference sentences. Metric score is calculated based on the modified reference and candidate sentences. A missing word penalty and a penalty for limited reference sentences is added (or subtracted) from the modified metric score. It should be noted that the penalty is subtracted from the modified metric score if the lower metric score shows better performance (e.g., WER and TER.).

deletion cost = 1 and substitution cost = 2. To address $R2$, we decrease the substitution cost to ρ_{sub} for similar-sounding characters as substitution of one of these characters is highly likely. To address $R3$, we decrease the addition cost of vowels to ρ_{add} and the deletion cost of vowels to ρ_{del}, where $\rho_{add} > \rho_{del}$. This is due to the empirical observation that the omission of vowels is much more likely than an addition. Further, we decrease the addition and deletion costs of a possible silent character to ρ_{sil}. We consider the minimum of PDS($w1$, $w2$) and PDS($w2$, $w1$) as the final PDS score to identify the spelling variation between words $w1$ and $w2$. In our experiments, we keep $\rho_{sub} = \rho_{add} = \rho_{sil} = 0.75$, and $\rho_{del} = 0.25$.

3.2 Similar Words

Identifying similar words in the same or different languages is a challenging task in the CM languages. For example, two phrases *"in the market"* and *"in the bazaar"* convey the same semantics, but most automatic evaluation metrics will fail to identify the semantic similarity. To address the challenge of token-level similarity, we need a common representation of words across the source languages. To mitigate this problem, we propose a Similar Word Search (SWS) procedure. Algorithm 2 shows the description of the SWS procedure. Given a word from the candidate CM sentence as an input, the SWS procedure returns all similar words from the corresponding reference sentences. We select that word from the reference sentences, which yields the minimum PDS value. The SWS procedure returns a word from the reference set if the minimum PDS value is less than σ_{thres}. Otherwise, it computes pairwise cosine distance (in the cross-lingual word embedding space) between each word in a set of reference words and the input word. To create the cross-lingual embedding space, we use the pre-trained word vectors of dimension 300 for English and Hindi from fastText (Bojanowski et al., 2017). For the shared representation, we use VecMap (Artetxe et al., 2018) to learn the mapping in an unsupervised fashion with the default settings. We use the English and Hindi sentences from the IIT-B parallel corpus (Kunchukuttan et al., 2018). In case the cosine similarity is greater than σ_{cos}, the SWS procedure returns the word from the reference set; else, we assume that no similar word exists in the reference set. In our experiments, we keep $\sigma_{thres} = 2$, and $\sigma_{cos} = 0.5$.

3.3 Missing words

Generally, the generated candidate sentence misses some words resulting in a significant impact on the automatic evaluation scores. Some words are more important than others, but most

Algorithm 2: Similar Word Search

Def SWS(*references, word, distance*=σ_{thres}):
- R=$\{r_1, r_2, ..., r_m\}$ be the set of sentences in *references*. Covert each r_i to lowercase and remove all special characters.
- Make a set W of words for each r_i in R.
- $PDS_{min} \leftarrow distance$, $cand \leftarrow None$
- **for** *each word* w_i *in* W **do**
  - **if** $PDS(w_i, word) <= PDS_{min}$ **then**
    - $PDS_{min} \leftarrow PDS(w_i, word)$
    - $cand \leftarrow w_i$
- **if** *cand != None* **then**
  - **return** cand
- **else**
  - word_emb ← CLWE(word) , $max_{sim} \leftarrow \sigma_{cos}$
  - **for** *each word* w_i *in* W **do**
    - sim ← cos_sim(word_emb, CLWE(w_i))
    - **if** *sim* >= max_{sim} **then**
      - $max_{sim} \leftarrow sim$, $cand \leftarrow w_i$
  - **if** *candidate != None* **then**
    - **return** cand
  - **else**
    - **return** word

metrics consider them equal ($M1$). Furthermore, most metrics match exact words with no flexibility in spelling variations and language switching ($M2$). Here, we address both these problems to apply a missing word penalty to the metric score with some writing style flexibility. To address $M1$, we use WAC procedure[2] to generate a large Hinglish corpus (hereafter *'ParallelCorp'*) of 2,132,184 sentences. For creating the parallel corpus, we collect English sentences from multiple sources[3,4,5,6] and translate them (if not already translated) into Hindi language using Google Translate API. We calculate IDF-values (Inverse Document Frequency) of each word in the Hinglish corpus. The words with low IDF values occur rarely and hence carry more semantic information. If a word is not present in the *ParallelCorp*, we consider it semantically important. To address $M2$, we relax the exact match condition by postulating that either the word is present in the candidate sentence or its variant is present in the sentence. Here, we allow two types of variations (i) minor spelling variations and (ii) language switch (for more details, see Sections 3.1 and 3.2). We use the SWS procedure to find a word variant keeping a maximum distance value of 1. Algorithm 3 shows the description of the missing word penalty (MWP) in detail. For each word w in a reference sentence, we check the presence of w and its variants in the candidate sentence. In case w is not present, we add w's IDF value as the penalty for the absence. We repeat the procedure for each reference sentence and take the minimum penalty among all reference sentences. We reduce the MWP score from the metric score for a given evaluation metric.

Algorithm 3: Missing Words Penalty

Def MWPenalty(*references, sentence*):
- Set min_penalty ← 1
- **for** *each* r_i *in references* **do**
  - Set penalty ← 0
  - Set total_imp ← 0
  - Create a set W of words in r_i
  - **for** *each word* w_i *in* W **do**
    - total_imp += IDF(w_i)
    - **if** w_i *not in sentence and* $SWS([sentence], w_i, 1) == w_i$ **then**
      - penalty += IDF(w_i)
  - penalty /= total_imp
  - **if** *penalty* < *min_penalty* **then**
    - min_penalty ← penalty
- **return** min_penalty

3.4 Limited Reference Sentences

A sentence can be paraphrased in numerous ways by interchanging subject and predicate, active and passive voice, and first, second, and third-person perspectives. With the code-mixed text, the paraphrasing possibilities significantly increase. For an automatic evaluation, it is infeasible to generate all possible paraphrases as reference sentences. Even though *HinGE* dataset has at least two reference sentences against a candidate sentence but it is insufficient to include all the possibilities. Thus, paraphrasing drastically limits the evaluation capabilities of various metrics. To address this problem, we present an algorithm *PhraseScore*. Algorithm 4 shows the description of the *PhraseScore* method. We split the candidate sentence and the set of reference sentences into trigram phrases. If word w in the candidate phrase exists in one of

[2]We employ WAC due to its capability to generate high-quality sentences (as shown in (Srivastava and Singh, 2021b)). Also, the Hinglish sentence generated by WAC has words from only the source English and Hindi sentences which in turn doesn't influence the IDF values of the generated words to a large extent.

[3]https://www.kaggle.com/kazanova/sentiment140

[4]https://www.kaggle.com/arkhoshghalb/twitter-sentiment-analysis-hatred-speech

[5]http://www.cfilt.iitb.ac.in/iitb_parallel/

[6]https://bit.ly/2XQjrU6

Algorithm 1: Phonetic Dissimilarity

Def PDS(w_1, w_2):
- Remove consecutive duplicate characters from both the words w_1 and w_2
- Set dp[len(w_1)+1][len(w_2)+1] = {0}
- **for** *i* ← *0 to len(w_1)* **do**
  - **for** *j* ← *0 to len(w_2)* **do**
    - **if** *i == 0* **then**
      - dp[i][j] ← j
      - continue
    - **if** *j == 0* **then**
      - dp[i][j] ← i
      - continue
    - **if** w_1*[i-1]* == w_2*[j-1]* **then**
      - dp[i][j] ← dp[i-1][j-1]
    - **else**
      - sub ← 2, add ← 1, del ← 1
      - **if** w_1*[i-1] and* w_2*[j-1] are similar sounding* **then**
        - sub ← ρ_{sub}
      - **if** w_1*[i-1] can be silent* **then**
        - delete ← ρ_{sil}
      - **if** w_1*[i-1] is a vowel* **then**
        - delete ← ρ_{del}
      - **if** w_2*[j-1] can be silent or a vowel* **then**
        - add ← ρ_{add}
      - dp[i][j] ← min(dp[i][j-1]+add, dp[i-1][j]+del, dp[i-1][j-1]+sub)
- Return dp[len(w_1)][len(w_2)]

the reference phrases, we add the IDF value of the w in the phrase score for that phrase. Else, we subtract the IDF value as a penalty. This phrase score is aggregated, normalized over the number of phrases in the candidate sentence, and divided by the penalty of missing words in the candidate sentence. To prevent division by zero, we add 0.0001 to the penalty. In case a word is not present in the IDF dictionary, we assign it a relatively high value (μ_{miss}) to indicate that it is a rare word of high importance. Finally, we increase the metric score by adding the candidate sentence's *PhraseScore*. In our experiments, we keep μ_{miss}=20. Due to the unavailability of a paraphrasing system for a code-mixed language, the formulation of *PhraseScore* algorithm depends on the assumption that the trigram phrases in a sentence can be reordered to create new sentences.

4 Results and Evaluation

We evaluate WAC and PAC procedures augmented with *MIPE* pipeline against all the five metrics (as discussed in Section 2). Table 1 and 2 shows the effect of *MIPE* against the five metrics. As expected, all metrics show better scores with the *MIPE* augmentation. The metric scores after the *MIPE* shows a high correlation[7] against the metric scores without *MIPE* (see Table 3). This shows that improvements in the metric scores is constant throughout and are not by chance.

Algorithm 4: Limited Reference Sentences

Def PhraseScore(*references, candidate, penalty, idf*):
- Let $R=\{r_1, r_2, ..., r_m\}$ be the set of sentences in *references*. Tokenize the *candidate* and the *references* and convert it into a list of phrases of 3 tokens. Let the lists be $sent_{phrase}$ and ref_{phrase} respectively.
- Set $sent_{score}$ ← 0
- **for** *each* s_{phrase} *in* $sent_{phrase}$ **do**
  - $score$ ← 0
  - **for** *each* r_{phrase} *in* ref_{phrase} **do**
    - t_{score} ← 0
    - **for** *each word* w_i *in* s_{phrase} **do**
      - **if** w_i *in* r_{phrase} **then**
        - **if** w_i *in* idf*.keys()* **then**
          - t_{score} += $idf[w_i]$
        - **else**
          - t_{score} += μ_{miss}
      - **else**
        - **if** w_i *in* idf*.keys()* **then**
          - t_{score} -= $idf[w_i]$
        - **else**
          - t_{score} -= μ_{miss}
    - $score$ = max($score$, t_{score})
  - $sent_{score}$ += $score$
- $sent_{score}$ = $sent_{score}$ / (len($sent_{phrase}$) * (penalty + 0.0001))
- **return** $sent_{score}$

Table 4 shows the effect of *MIPE* on the correlation with the human scores. We use the same criteria to bucket the human ratings as discussed in Section 2. We observe a higher correlation in all the three buckets for WAC augmented with *MIPE*. This improvement is consistent throughout all the metrics. For PAC augmented with *MIPE*, we observe a decrease in correlation in the second bucket, which can be attributed to (i) a relatively large number of poor quality (low human scores) sentences generated by PAC, and (ii) rating poor quality CM sentence is a challenging task for humans due to lower readability of the sentence. For the rest of the buckets, PAC with *MIPE* shows a higher correlation with the human scores.

5 Current State and Future Directions

The results discussed in Section 4 demonstrate a need to build metrics, theories, and experiments for better CM NLG evaluation. Some of the

[7] We experiment with Pearson Correlation Coefficient.

Human score	Without *MIPE*					With *MIPE*				
	BLEU	**WER**	**TER**	**NIST**	**BS**	**BLEU**	**WER**	**TER**	**NIST**	**BS**
2	0.144	0.741	0.667	0.092	0.851	0.238	0.651	0.544	0.140	0.860
3	0.138	0.735	0.708	0.070	0.852	0.323	0.625	0.569	0.133	0.860
4	0.133	0.695	0.666	0.103	0.849	0.391	0.536	0.480	0.184	0.906
5	0.135	0.711	0.681	0.110	0.853	0.380	0.556	0.494	0.172	0.985
6	0.141	0.697	0.670	0.102	0.852	0.361	0.560	0.502	0.144	0.967
7	0.161	0.663	0.630	0.111	0.856	0.398	0.522	0.453	0.168	0.947
8	0.177	0.621	0.589	0.127	0.859	0.465	0.445	0.377	0.204	0.976
9	0.212	0.571	0.538	0.150	0.865	0.531	0.387	0.313	0.242	1.000
10	0.291	0.509	0.493	0.157	0.878	0.572	0.318	0.252	0.291	1.000

Table 1: Comparison of metric scores with and without using *MIPE* for WAC.

Human score	Without *MIPE*					With *MIPE*				
	BLEU	**WER**	**TER**	**NIST**	**BS**	**BLEU**	**WER**	**TER**	**NIST**	**BS**
2	0.126	0.672	0.698	0.176	0.8603	0.338	0.474	0.500	0.318	0.997
3	0.146	0.765	0.696	0.086	0.851	0.425	0.603	0.526	0.120	0.883
4	0.143	0.744	0.703	0.100	0.8464	0.419	0.598	0.523	0.130	0.888
5	0.153	0.726	0.680	0.114	0.8515	0.407	0.589	0.508	0.154	0.894
6	0.164	0.689	0.646	0.124	0.8558	0.449	0.525	0.456	0.171	0.912
7	0.176	0.661	0.618	0.121	0.8581	0.475	0.485	0.411	0.198	0.936
8	0.177	0.639	0.605	0.128	0.8598	0.498	0.437	0.370	0.200	0.938
9	0.184	0.614	0.590	0.129	0.8638	0.545	0.387	0.321	0.230	0.967
10	0.242	0.551	0.543	0.146	0.8731	0.600	0.314	0.262	0.280	0.997

Table 2: Comparison of metric scores with and without using *MIPE* for PAC.

	BLEU	**WER**	**TER**	**NIST**	**BS**
WAC	0.948	0.988	0.984	0.961	0.8326
PAC	0.830	0.986	0.982	0.944	0.8843

Table 3: Correlation between the evaluation metric scores with and without using *MIPE* pipeline.

	Correlation with human scores (Without MIPE)						**Correlation with human scores (With MIPE)**					
	Bucket 1		**Bucket 2**		**Bucket 3**		**Bucket 1**		**Bucket 2**		**Bucket 3**	
	WAC	**PAC**	**WAC**	**PAC**	**WAC**	**PAC**	**WAC**	**PAC**	**WAC**	**PAC**	**WAC**	**PAC**
BLEU	0.810	0.910	-0.861	**0.878**	0.941	0.844	**0.942**	**0.950**	**0.910**	0.643	**0.994**	**0.981**
WER	-0.936	**-0.822**	-0.785	**0.457**	-0.993	-0.973	**-0.949**	-0.780	**-0.880**	0.713	**-0.995**	**-0.993**
TER	-0.891	**-0.963**	0.000	**-0.610**	-0.998	-0.970	**-0.932**	-0.937	**-0.737**	0.229	**-0.998**	**-0.997**
NIST	**0.913**	0.127	0.642	**-0.559**	0.986	0.846	0.851	**0.246**	**0.769**	-0.671	**0.993**	**0.952**
BS	0.844	**0.710**	0.227	**-0.689**	**0.953**	0.937	**0.924**	0.400	**0.922**	-0.720	0.895	**0.972**

Table 4: Comparison of correlation between evaluation metrics and human scores for WAC and PAC with and without *MIPE* pipeline. The bold numbers indicate a better correlation in the respective bucket. A high positive correlation is preferred for BLEU and NIST whereas a negative correlation is preferred for WER and TER.

challenges and limitations of the proposed *MIPE* pipeline for effective CM NLG includes:

- Due to the unavailability of resources in other CM language pairs, the *MIPE* pipeline is tested on a single CM language. We need to extend the proposed evaluation strategy to other CM language pairs.
- The presence of two different languages in a single CM sentence increases the paraphrasing possibility to a much larger extent. We need metrics that attend to the CM sentences beyond the bag of words model. These metrics should also be able to account for paraphrasing.
- There are various other reasons (beyond the four reasons discussed in this paper) that influence the evaluation of CM NLG tasks such as named-entities, transliteration, etc. The *MIPE* pipeline doesn't currently account for these limitations.

- The code-mixed sentences in the *HinGE* dataset are not collected from the social media platforms. The code-mixed data from the social media platform tends to be more noisy and distorted which could influence the performance of *MIPE* pipeline.

As discussed, currently there are several limitations with the CM NLG evaluation which need to be addressed in order to build effective CM NLG systems for multilingual societies. Some of the lessons learned and the future directions for the CM NLG evaluations are:

- The limited resource availability is one of the major bottlenecks in the CM NLG tasks and evaluation. Currently, the available resources are smaller in size compared to the monolingual NLG tasks.
- In contrast to the *MIPE* augmentation pipeline, we need systems that can leverage the noisy nature of the code-mixed text. The currently proposed *MIPE* pipeline addresses the various challenges independently and attempts to reconstruct the noisy CM text for effective evaluation.
- The two languages participating in CM influence the various constructs of the target CM sentence such as grammar, syntax, etc. The current experimentation with only one CM language needs to be explored with other CM languages.
- Recently, we observe a rise in the availability of multilingual language models (LMs). These LMs could be used to build effective CM NLG evaluation systems.
- The current evaluation metrics seem to perform poorly with the CM languages. We need to build dedicated metrics for the CM NLG evaluation tasks that can leverage the linguistic diversity of the CM data.

6 Conclusion

In this paper, we present a metric independent evaluation pipeline for efficient code-mixed NLG evaluation. The proposed pipeline shows a high correlation between the human scores and the underlying evaluation metrics. Besides the four significant challenges to CM NLG evaluation, in the future, we also plan to address other challenges such as code-mixed existence of named-entities, informal writing style, and missing context.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Scott Baldauf. 2004. A hindi-english jumble, spoken by 350 million. [Online; accessed 23-May-2020].

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the first workshop on computational approaches to code switching*, pages 13–23.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Khyathi Chandu, Ekaterina Loginova, Vishal Gupta, Josef van Genabith, Günter Neumann, Manoj Chinnakotla, Eric Nyberg, and Alan W Black. 2019. Code-mixed question answering challenge: Crowdsourcing data and techniques. In *Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 29–38. Association for Computational Linguistics (ACL).

Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. 2018. 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.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Devansh Gautam, Prashant Kodali, Kshitij Gupta, Anmol Goel, Manish Shrivastava, and Ponnurangam Kumaraguru. 2021. 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.

Sunil Gundapu and Radhika Mamidi. 2018. Word level language identification in english telugu code mixed data. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*.

Abhirut Gupta, Aditya Vavre, and Sunita Sarawagi. 2021. 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.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2267–2280.

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

Ganesh Jawahar, Muhammad Abdul-Mageed, VS Laks Lakshmanan, et al. 2021. Exploring text-to-text transformers for english to hinglish machine translation with synthetic code-mixing. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 36–46.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

Simran Khanuja, Sandipan Dandapat, Sunayana Sitaram, and Monojit Choudhury. 2020. A new dataset for natural language inference from code-mixed conversations. In *Proceedings of the The 4th Workshop on Computational Approaches to Code Switching*, pages 9–16.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The iit bombay english-hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Dwija Parikh and Thamar Solorio. 2021. Normalization and back-transliteration for code-switched data. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 119–124.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, PYKL Srinivas, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 774–790.

Adithya Pratapa, Monojit Choudhury, and Sunayana Sitaram. 2018. Word embeddings for code-mixed language processing. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3067–3072.

Ruba Priyadharshini, Bharathi Raja Chakravarthi, Mani Vegupatti, and John P McCrae. 2020. Named entity recognition for code-mixed indian corpus using meta embedding. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 68–72. IEEE.

Vikram Ramanarayanan, Robert Pugh, Yao Qian, and David Suendermann-Oeft. 2019. Automatic turn-level language identification for code-switched spanish–english dialog. In *9th International Workshop on Spoken Dialogue System Technology*, pages 51–61. Springer.

Mohd Sanad Zaki Rizvi, Anirudh Srinivasan, Tanuja Ganu, Monojit Choudhury, and Sunayana Sitaram. 2021. 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.

Sushmitha Reddy Sane, Suraj Tripathi, Koushik Reddy Sane, and Radhika Mamidi. 2019. 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.

Shashi Shekhar, Dilip Kumar Sharma, and MM Sufyan Beg. 2020. Language identification framework in code-mixed social media text based on quantum lstm—the word belongs to which language? *Modern Physics Letters B*, 34(06):2050086.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018a. Language identification and named entity recognition in hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018b. A twitter corpus for hindi-english code mixed pos tagging. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 12–17.

Rajat Singh, Nurendra Choudhary, and Manish Shrivastava. 2018c. Automatic normalization of word variations in code-mixed social media text. *arXiv preprint arXiv:1804.00804*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200. Cambridge, MA.

Vivek Srivastava and Mayank Singh. 2020. Phinc: a parallel hinglish social media code-mixed corpus for machine translation. *arXiv preprint arXiv:2004.09447*.

Vivek Srivastava and Mayank Singh. 2021a. 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.

Vivek Srivastava and Mayank Singh. 2021b. Hinge: A dataset for generation and evaluation of code-mixed hinglish text. *arXiv preprint arXiv:2107.03760*.

S Thara, E Sampath, Phanindra Reddy, et al. 2020. Code mixed question answering challenge using deep learning methods. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 1331–1337. IEEE.

Kristina Toutanova and Robert C Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151.

Jethva Utsav, Dhaiwat Kabaria, Ribhu Vajpeyi, Mohit Mina, and Vivek Srivastava. 2020. Stance detection in hindi-english code-mixed data. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, CoDS COMAD 2020, page 359–360, New York, NY, USA. Association for Computing Machinery.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.

Genta Indra Winata, Zhaojiang Lin, and Pascale Fung. 2019. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020. Csp: Code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624–2636.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.