Co-BERT: A Comprehensive Evaluation of Tokenization Techniques in Code-Switching Environments

Madhan S[1], Gnanesh A R[1], Gopal[1] and **Bharath L[1]**
[1]Department of Data Science and Artificial Intelligence,
(22bds036, 22bds023, 22bds025, 22bds013)@iiitdwd.ac.in

Abstract

Code-mixed languages, characterized by the blending of two or more languages in a single utterance, pose significant challenges in natural language processing (NLP). Tokenization plays a crucial role in handling such linguistically complex data effectively. In this study, we comprehensively evaluate three tokenization techniques—WordPiece, SentencePiece, and Byte Pair Encoding (BPE)—across three distinct code-mixed languages: Tanglish, Kanglish, and Hinglish. By leveraging a dataset collected from YouTube video transcriptions and transliterated using AI4Bharat's Bhashini, we systematically determine the optimal tokenizer for each language based on loss function analysis. The results provide nuanced insights into the efficiency of different tokenization strategies for diverse linguistic structures in code-mixed text, contributing to the advancement of multilingual NLP research.

1 Introduction

Code-mixing is a phenomenon in which speakers mix words, phrases, and clauses from multiple languages within a single conversation or sentence. The intricate nature of linguistic interaction in such contexts has long fascinated researchers, revealing the sophisticated communicative strategies employed by multilingual speakers (Gumperz, 1982). As digital communication platforms continue to expand, the need for advanced natural language processing (NLP) techniques capable of handling mixed-language data has become increasingly critical (Pasad et al., 2022). The exponential growth of online content and social media has dramatically transformed the landscape of linguistic communication.

In the Indian context, this transformation is particularly pronounced, and speakers seamlessly navigate between regional languages and English, creating rich and complex linguistic tapestries (Joshi et al., 2016). Our research addresses this critical gap by developing a specialized code-mixed language model specifically tailored to Indian languages. By focusing on Tamil, Kannada and Hindi, our objective is to create a comprehensive approach that can effectively process and understand the intricate linguistic patterns unique to these languages.

Technologically, code-mixed language processing represents a frontier in natural language processing (NLP) that demands innovative computational approaches. Traditional monolingual models fail to capture the intricate linguistic interactions that occur when languages intermingle at the lexical, syntactic, and pragmatic levels. This limitation becomes particularly pronounced on digital communication platforms, where code mixing has become an increasingly prevalent mode of expression.The study employs a sophisticated methodology that comparatively evaluates three advanced tokenization techniques: Byte Pair Encoding (BPE) (Sennrich et al., 2016), WordPiece (WP) (Wu et al., 2016) and SentencePiece (SP) (Kudo and Richardson, 2018). The complete codebase for our system is available at https://github.com/madhans476/Co-BERT.git.

2 Literature Review

The emergence of transformer-based models has revolutionized natural language processing, with the BERT (Bidirectional Encoder Representations from Transformers) model by Devlin et al. (Devlin et al., 2018) standing as a landmark contribution. BERT introduced a groundbreaking approach to pre-training language representations, utilizing masked language modeling and next sentence prediction tasks to create contextually rich embeddings. The model's ability to capture bidirectional context marked a significant advancement over previous unidirectional language models.

In the context of multilingual and code-mixed language processing, the BERT architecture has provided a robust foundation for advanced NLP techniques. The model's transformer-based approach, with its self-attention mechanism, offers unprecedented capabilities in understanding complex linguistic contexts. Researchers have increasingly adapted the BERT framework to address specific linguistic challenges, particularly in multilingual environments. Our research

builds directly upon the foundational work presented in the BERT paper, leveraging its architectural insights to develop a specialized approach for code-mixed Indian languages. By focusing on the core transformer architecture, we aim to demonstrate the model's adaptability to linguistically diverse and complex communication contexts.

3 Dataset

Code-Mix Version	Unique words count
Tanglish	5995
Kanglish	5867
Hinglish	5847

Table 1: Unique Word Counts in Code-Mixed Versions

3.1 Data Source

Our research leveraged YouTube video transcripts as the primary source of linguistic data for three Indian languages: Tamil, Hindi, and Kannada. YouTube was selected due to its rich, diverse, and conversational content that naturally reflects code-mixed language usage (Joshi et al., 2016). The platform provides authentic, user-generated content that captures the nuanced linguistic interactions typical of multilingual speakers.

3.2 Transliteration Process

To standardize the data and facilitate computational analysis, we employed AI4Bharat's(Antony et al., 2021) Bhashini transliteration tool (Madhani et al., 2022), as well as MistralAI's language model(Jiang et al., 2023) for transliteration. These advanced systems convert the original language scripts to a romanized representation, enabling a uniform approach to processing code-mixed text.

4 Methodology

This methodology section provides a comprehensive overview of the approaches and techniques employed in processing and analyzing code-mixed data. It encompasses two primary components: tokenization strategies and model architecture design. By carefully selecting and implementing appropriate tokenization methods and leveraging the robust BERT architecture, we aim to effectively handle the complexities inherent in code-mixed linguistic data.

4.1 Tokenizers

Tokenizers are fundamental preprocessing tools in natural language processing (NLP) that break down text into smaller units called tokens. These tokens can be words, subwords, or characters, which are then used as input for machine learning models. Tokenization is crucial for handling different languages, especially in code-mixing scenarios, as it helps manage vocabulary and reduce out-of-vocabulary issues.

4.1.1 BytePair Encoding Tokenizer

BytePair Encoding (Sennrich et al., 2016) (BPE) is a powerful tokenization algorithm that iteratively merges the most frequent character pairs in a given corpus. The algorithm works as follows:

- Initially, the tokenizer starts by splitting words into individual characters.
- It then identifies the most frequent pair of characters or subwords in the training data.
- These frequent pairs are merged into a new token, creating a more compact representation.
- The process is repeated for a predefined number of iterations or until a target vocabulary size is reached.

BPE is particularly effective for handling morphologically rich languages and reducing vocabulary size while maintaining the ability to represent rare or out-of-vocabulary words through subword tokenization.

4.1.2 WordPiece Tokenizer

The WordPiece tokenizer (Wu et al., 2016) (WP) is similar to BPE but with a slightly different approach to token generation:

- It begins by splitting text into individual characters.
- The algorithm selects token merges based on a likelihood criterion, maximizing the likelihood of the training data.
- Instead of simply choosing the most frequent pair, WordPiece considers the probability of the merged token in the context of the entire corpus.
- It creates a vocabulary of subwords that can effectively represent words while managing out-of-vocabulary challenges.

WordPiece was notably used in the original BERT (Devlin et al., 2018) model and is well-suited for handling diverse linguistic variations.

4.1.3 SentencePiece Tokenizer

SentencePiece (Kudo and Richardson, 2018) (SP) is a language-independent tokenizer that provides a unique approach to text segmentation:

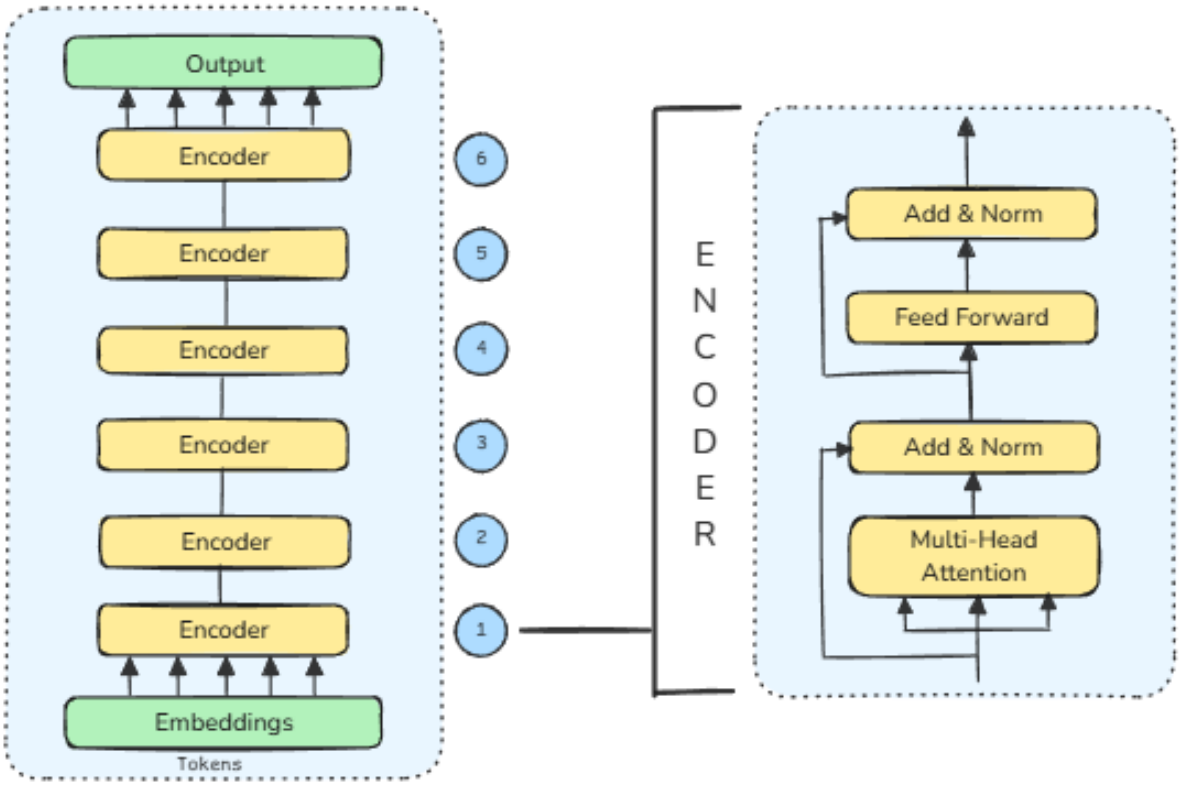

Figure 1: Co-BERT architecture with 6 encoder layers (left) and detailed structure of a single encoder block (right). The model incorporates token embeddings at the bottom, passes through 6 sequential transformer encoder layers, and produces contextual representations at the output layer.

- It treats the input as a sequence of Unicode characters.
- The algorithm can perform unsupervised tokenization using either BPE or Unigram Language Model algorithms.
- It handles multiple languages without requiring language-specific preprocessing like word segmentation.
- SentencePiece is particularly useful for multilingual models and code-mixed text, as it can tokenize text without relying on language-specific word boundaries.

Key advantages include its ability to handle various writing systems and its language-agnostic nature.

4.2 Model Architecture

Our implementation as shown in figure 1, adapts the BERT (Devlin et al., 2018)architecture with the following key components:

- **Embedding Layer**:
  - Token embeddings with vocabulary size 2000
  - Position embeddings (max length 128)
  - Segment embeddings for sentence pairs
  - Layer normalization and dropout (p=0.1)
- **Transformer Encoder**:
  - 6 layers (vs original BERT's 12)
  - Hidden size 768, 12 attention heads
  - Intermediate size 3072 (4x hidden size)
  - **Activation Function:** GELU (Gaussian Error Linear Unit)

Hyper parameter	Value
Learning Rate	5×10^{-5}
Weight decay	0.01
ϵ	1×10^{-8}
dropout	0.1
Gradient clipping	1.0

Table 2: Hyper parameters

- **Loss Function**:
  - Cross Entropy loss
  - Weighted combination of objectives:

$$\mathcal{L}_{total} = \mathcal{L}_{MLM} + \mathcal{L}_{NSP} \quad (1)$$

Where:
$\mathcal{L}_{MLM}$ = Masked Language Model loss
$\mathcal{L}_{NSP}$ = Next Sentence Prediction loss
- Used AdamW optimizer.

4.2.1 Pre-training Heads

- Masked Language Model (MLM):
  - 15% random token masking
  - 80% replaced with [MASK]
  - 10% replaced with random token
  - 10% kept unchanged
- Next Sentence Prediction (NSP):
  - Binary classification head
  - 50% positive (actual next sentences)
  - 50% negative (random sentences)

5 Results

Our comparative analysis of tokenization techniques across different code-mixed languages reveals nuanced performance variations. Table 3 illustrates the loss function scores for three tokenization approaches: BPE, WordPiece (WP), and SentencePiece (SP).

Tokenizer	Tanglish	Kanglish	Hinglish
BPE	7.20	7.32	7.18
WordPiece	7.43	7.42	7.06
SentencePiece	7.73	7.73	7.56

Table 3: Validation loss across tokenizers and language pairs

5.1 Analysis

Our comparative analysis reveals distinct patterns across tokenization techniques:

- **Hinglish**: WordPiece demonstrates superior performance with the lowest loss of 7.06, outperforming BPE (7.18) and SentencePiece (7.56).
- **Tanglish**: BPE achieves the lowest loss of 7.20, followed by WordPiece (7.43) and SentencePiece (7.73).
- **Kanglish**: BPE slightly outperforms WordPiece with a loss of 7.32 compared to 7.42, while SentencePiece trails with 7.73.

The results indicate that the optimal tokenization strategy varies by language pair. WordPiece demonstrates superior performance for Hinglish and competitive performance for Kanglish, while BPE excels in Tanglish contexts. SentencePiece consistently shows higher loss values across all language pairs. The performance differentials between tokenization techniques, ranging from 0.07 to 0.57 loss points, highlight the significance of selecting appropriate tokenization strategies for specific code-mixed language contexts.

6 Conclusion

This research provides critical insights into the effectiveness of tokenization techniques for processing code-mixed Indian languages. Our comprehensive analysis of BPE, WordPiece, and SentencePiece across Tanglish, Kanglish, and Hinglish reveals several key findings:

- Different tokenization methods demonstrate variable efficacy across language pairs, suggesting language-specific optimization may be necessary.
- WordPiece demonstrates superior performance for Hinglish data, while BPE excels in Tanglish and Kanglish contexts.
- The performance variations between tokenization techniques were substantial in some cases, with loss differences ranging from 0.07 to 0.57.
- SentencePiece consistently demonstrated higher loss values across all language pairs, suggesting potential limitations in handling code-mixed Indian languages.

The methods and findings presented in this work have broad applications in:

- Multilingual social media analysis
- Code-mixed conversational AI systems
- Digital content moderation for Indian languages
- Language-specific tokenization strategies for mixed-language contexts

Our approach demonstrates that language-specific tokenization strategies combined with architectural modifications can significantly improve code-mixed language processing, paving the way for more inclusive NLP systems in multilingual societies. The results underscore the importance of tailoring preprocessing techniques to specific linguistic contexts rather than applying a one-size-fits-all approach to code-mixed language processing.

References

J. Antony et al. 2021. Ai4bharat: A benchmark for indian languages. *arXiv preprint arXiv:2103.03055*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*.

John J. Gumperz. 1982. *Discourse strategies*. Cambridge University Press, Cambridge.

Yao Jiang, Guillaume Lample, Sebastian Ruder, Angela Fan, and Myle Ott. 2023. Mistral: Fast and open foundation models. Accessed: 2025-04-22.

P. Joshi et al. 2016. Towards code-mixing: Where linguistics meet technology. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Yash Madhani, Sushane Parthan, Priyanka A. Bedekar, Ruchi Khapra, Vivek Seshadri, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M. Khapra. 2022. Aksharantar: Towards building open transliteration tools for the next billion users. *ArXiv*, abs/2205.03018.

A. Pasad et al. 2022. Evaluation of multilingual nlp models. *Proceedings of the 2022 Conference on Natural Language Processing*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Jingbo Chen, Xian Zhu, Wei Wang, and et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1535–1545, Austin, Texas. Association for Computational Linguistics.