Comparative Analysis of Tokenizers in Tamil Text Classification in Low Resource Settings

Gokulan Sivakumaran[1], **Randil Pushpananda**[2], **ERAD Bandara**[1]
[1]Department of Statistics, University of Colombo, Sri Lanka
[2]University of Colombo School of Computing, University of Colombo, Sri Lanka
sivakumarangokulan@gmail.com, rpn@ucsc.cmb.ac.lk, anjana@stat.cmb.ac.lk

Abstract

Tokenization is crucial in NLP, influencing performance for morphologically rich, low resource languages like Tamil. This study comprehensively analyzes WordPiece, SentencePiece, and Byte-Level Byte Pair Encoding (BBPE) for Tamil text classification. We assess tokenization efficiency using metrics including token count, fragmentation, OOV rate, and compression ratio. Additionally, we analyze downstream impact through Tamil news title classification using a custom lightweight BERT based Transformer architecture. Tokenizers were pretrained on a 5.45 GB Tamil Corpus and evaluated on a Kaggle Tamil News Dataset. Results indicate WordPiece and SentencePiece outperform BBPE in efficiency and accuracy. While BBPE eliminates OOV words, excessive fragmentation hinders model learning. Increasing vocabulary size improves WordPiece and SentencePiece but not BBPE. Misclassification analysis highlights overfragmentation challenges. This study contributes to Tamil NLP by comparing tokenizers, aiding researchers in selecting appropriate strategies for agglutinative languages.

Keywords: Tokenization, Tamil, subword methods, low resource NLP, text classification

1. Introduction

Tokenization is a critical component of modern Natural Language Processing (NLP) systems, directly affecting vocabulary efficiency, semantic representation, and downstream task performance in transformer based models (Devlin et al., 2019; Wolf et al., 2020). Subword tokenization methods such as WordPiece, Byte Pair Encoding (BPE), and SentencePiece have become standard in high resource language settings (Wu et al., 2016; Sennrich et al., 2016; Kudo and Richardson, 2018a). These approaches effectively mitigate large vocabulary sizes and Out Of Vocabulary (OOV) issues while preserving compositional semantics.

However, their effectiveness in morphologically rich and low resource languages remains insufficiently understood. Tamil, an agglutinative Dravidian language, presents substantial challenges due to its extensive inflectional morphology, compound formation, and divergence between written and spoken forms (Annamalai and Asher, 2002; Thangarasu and Manavalan, 2013; Sarveswaran et al., 2021). When subword tokenizers designed primarily for English centric corpora are applied directly to Tamil, they often produce excessive fragmentation or linguistically inconsistent segmentations, potentially degrading semantic coherence and model performance.

Transformer architectures increasingly rely on subword tokenization to control vocabulary growth and improve generalization (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019). Prior research demonstrates that tokenization design can significantly impact representation quality and downstream performance (Bostrom and Durrett, 2020; Rust et al., 2020). Nevertheless, systematic evaluations across morphologically complex low resource languages remain limited. Existing Tamil NLP efforts largely emphasize preprocessing tools and rule based segmentation pipelines rather than empirical comparisons of modern subword tokenization strategies (Singh and Shah, 2022). Consequently, the relationship between tokenization granularity and downstream task performance in Tamil has not been rigorously examined.

To address this gap, we performed a comparative evaluation of three widely used subword tokenization methods: WordPiece, SentencePiece, and BBPE. We assess both intrinsic tokenization characteristics and extrinsic downstream performance using Tamil news title classification with custom lightweight BERT based Transformer architecture (Devlin et al., 2019). News headlines constitute a suitable benchmark because they are short, semantically dense, and require precise lexical segmentation for accurate categorization (Veeramanju, 2023).

This paper makes the following contributions:

- Assess and compare the tokenization efficiency of BBPE, WordPiece, and SentencePiece using various Tokenization Quality metrics.
- Investigate how effectively each tokenization approach preserves semantic meaning in Tamil text classification by analyzing the performance of the model on a downstream task, specifically news title classification.

Our findings highlight the importance of language

Proceedings of the Second workshop on Challenges in Processing South Asian Languages (CHiPSAL2026) @ LREC, pages 198–208
16 May 2026.

aware tokenization design and contribute to improving multilingual language modeling for underrepresented languages.

2. Related Work

2.1. Tokenization Strategies and Efficiency

The choice of tokenization algorithm critically impacts model performance, particularly in balancing semantic fidelity with computational efficiency. Qarah and Alsanoosay (2024) conducted a comparative analysis of WordPiece (WP), SentencePiece (SP), and Byte-Level BPE (BBPE) for Arabic NLP tasks. Their findings indicate that SP achieves stronger task performance by better modeling morphological variation, albeit at increased computational cost. In contrast, BBPE offers faster training and inference but suffers from semantic degradation due to excessive token fragmentation.

These trade-offs are further influenced by vocabulary size. Hiraoka et al. (2020) show that larger vocabularies improve representational capacity but increase memory and computation overhead, while Dagan et al. (2024) demonstrate that aggressive compression can limit semantic expressiveness. In low resource settings, where memory footprint and inference latency are constrained, selecting an appropriate tokenization strategy is therefore as critical as model architecture design.

2.2. Tokenization in Morphologically Rich Languages

For agglutinative and morphologically complex languages, subword tokenization has proven essential (Kudo and Richardson, 2018b; Rust et al., 2021). In Korean NLP, Park et al. (2020) demonstrate that a hybrid approach combining morphological analysis with subword segmentation (e.g., morpheme+BPE) often outperforms pure subword or pure morpheme tokenization across tasks including sentiment analysis.

Similarly, extensive research in Arabic NLP (Antoun et al., 2020; Abdul-Mageed et al., 2020; Abdelali et al., 2021) confirms that subword tokenization substantially improves performance across classification and sequence labeling tasks. However, these studies focus primarily on Semitic or Altaic language families.

Byte-level approaches, while language agnostic, introduce challenges in morphologically dense languages. Wei et al. (2021) show that byte-level subword modeling can generalize across scripts, but excessive fragmentation may obscure affixal and morphemic information, which is crucial for agglutinative languages such as those in the Dravidian family.

2.3. Research Gap

Despite extensive research on Arabic and Korean, there is a clear lack of systematic evaluations for Tamil, a low-resource and agglutinative Dravidian language. While foundational linguistic resources and NLP tools exist, few studies explicitly isolate the impact of tokenization strategies on both computational efficiency and downstream task performance.

Existing work often evaluates tokenization either through intrinsic efficiency metrics or downstream accuracy in isolation. This study addresses this gap by providing a holistic evaluation of WP, SP, and BBPE for Tamil, assessing intrinsic tokenization quality (e.g., fragmentation, compression, and vocabulary efficiency) alongside extrinsic performance on a news classification task. By establishing a unified evaluation framework, this work contributes toward more efficient and linguistically informed language models for underrepresented languages.

3. Background

This section introduces the core tokenization strategies and model architectures that form the foundation of this research. We briefly describe WordPiece, SentencePiece, and BBPE, followed by an overview of BERT based transformer models used for downstream evaluation.

3.1. WordPiece Tokenization

WordPiece is a subword tokenization algorithm originally proposed for neural machine translation and later widely adopted in transformer-based language models such as BERT (Wu et al., 2016; Devlin et al., 2019). The algorithm constructs a fixed size vocabulary by iteratively merging character sequences that maximize the likelihood of the training corpus. During tokenization, words are segmented into the longest matching subword units available in the learned vocabulary.

For example, the English sentence *"I was uncomfortable"* may be tokenized as: `[I]`, `[was]`, `[un]`, `[##comfortable]`, where the prefix `##` indicates that the token is a continuation of a preceding word. This strategy allows WordPiece to balance vocabulary coverage and semantic consistency while efficiently representing rare or unseen words through subword decomposition. The wordpiece tokenizer has been used in many prominent language models such as BERT, DistillBERT,and IndicBERT.

3.2. SentencePiece Tokenization

SentencePiece is an unsupervised and language independent subword tokenization framework that learns segmentation directly from raw text without requiring prior word level tokenization (Kudo and Richardson, 2018a; Kudo, 2018). Unlike traditional tokenizers that rely on whitespace splitting, SentencePiece treats input as a continuous stream of Unicode characters and learns subword units based on statistical properties of the training corpus.

For example, the English phrase *"natural language processing"* may be segmented into: `[_natural]`, `[_language]`, `[_process]`, `[ing]`, where the underscore symbol represents a whitespace boundary. SentencePiece supports both unigram language model tokenization and BPE based segmentation, allowing flexible handling of morphologically complex or multilingual text.The SentencePiece tokenizer has been used in many prominent language models such as ALBERT, mT5, XLNet , and T5.

3.3. Byte-Level Byte Pair Encoding (BBPE)

Byte-Level Byte Pair Encoding is a variant of the traditional BPE algorithm that operates directly on byte sequences rather than characters (Sennrich et al., 2016; Wei et al., 2021). By constructing a base vocabulary consisting of all possible byte values, BBPE guarantees complete text coverage and eliminates out-of-vocabulary tokens. We acknowledge that for languages like Tamil requiring multiple bytes per Unicode character in UTF-8 encoding, BBPE may introduce additional fragmentation compared to character-level BPE. However, BBPE was selected for this study for the following reasons:

1. **Standardization in Modern Architectures**: BBPE has become the default tokenization strategy in prominent transformer models such as GPT-2, RoBERTa, and GPT-3, making it a relevant baseline for contemporary NLP research (Radford et al., 2019; Liu et al., 2019).
2. **Cross-Lingual Comparability**: BBPE provides a language-agnostic approach that enables fair comparison across diverse scripts without requiring language-specific preprocessing (Wei et al., 2021).
3. **OOV Elimination**: The byte-level approach guarantees 0% OOV rate, which is valuable for low-resource settings where vocabulary coverage is critical.

3.4. BERT-based Transformer Models

Transformer-based language models rely on tokenized input sequences to learn contextual representations through self-attention mechanisms (Vaswani et al., 2017; Devlin et al., 2019). BERT and its variants employ bidirectional attention to capture contextual dependencies between tokens, making tokenization a critical component influencing model efficiency, representation quality, and downstream task performance.

Variants such as RoBERTa and ALBERT extend the original BERT architecture through improved training strategies and parameter efficiency (Liu et al., 2019; Lan et al., 2020). Since these models operate directly on token sequences, the choice of tokenization strategy significantly affects sequence length, semantic representation, and classification performance, particularly in morphologically rich languages.

4. Methodology

4.1. Software and Other Specifications

The experimental implementation was developed in Python, selected for its extensive ecosystem of machine learning libraries and strong community support. All experiments were executed on the High-Performance Computing (HPC) cluster to accommodate the substantial computational demands of model pretraining. The software environment was configured on Ubuntu 22.04 LTS. PyTorch as the primary deep learning frameworks for model development across pretraining and fine tuning stages. The Hugging Face ecosystem served as the cornerstone for NLP operations: the Transformers library provided model architectures (BertModel, BertForSequenceClassification), training utilities (Trainer, TrainingArguments), and tokenization (AutoTokenizer), while the Datasets library facilitated efficient data loading and management. It is important to note that

4.2. Model Pretraining

The objective of this study is to conduct a comparative analysis of three widely used subword tokenization strategies WordPiece, SentencePiece, and BBPE as base tokenizers for Tamil language models. The tokenizer pretraining was carried out using the Tamil Corpus Praveen Govi (2020) obtained from Kaggle.

The raw corpus underwent a preprocessing phase in which metadata, emojis, and non-Tamil scripts such as Hindi and Arabic were removed. Punctuation marks, numerals, symbols, and embedded English words were intentionally preserved to retain realistic linguistic structure. After preprocessing, the cleaned corpus size was approximately 5.45 GB.

The processed text corpus was loaded using the Hugging Face `datasets` library into a dataset

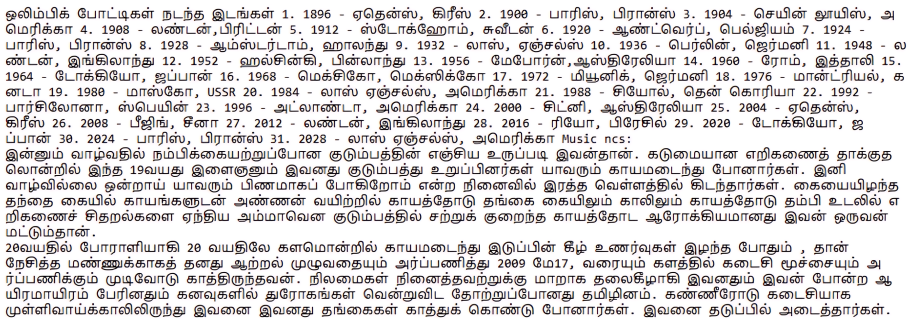
ஒலிம்பிக் போட்டிகள் நடந்த இடங்கள் 1. 1896 - ஏதென்ஸ், கிரீஸ் 2. 1900 - பாரிஸ், பிரான்ஸ் 3. 1904 - செயின் லூயிஸ், அ
மெரிக்கா 4. 1908 - லண்டன்,பிரிட்டன் 5. 1912 - ஸ்டோக்ஹோம், சுவீடன் 6. 1920 - ஆண்ட்வெர்ப், பெல்ஜியம் 7. 1924 -
பாரிஸ், பிரான்ஸ் 8. 1928 - ஆம்ஸ்டர்டாம், ஹாலந்து 9. 1932 - லாஸ், ஏஞ்சல்ஸ் 10. 1936 - பெர்லின், ஜெர்மனி 11. 1948 - ல
ண்டன், இங்கிலாந்து 12. 1952 - ஹல்சின்கி, பின்லாந்து 13. 1956 - மேபோர்ன்,ஆஸ்திரேலியா 14. 1960 - ரோம், இத்தாலி 15.
1964 - டோக்கியோ, ஜப்பான் 16. 1968 - மெக்சிகோ, மெக்ஸிக்கோ 17. 1972 - மியூனிக், ஜெர்மனி 18. 1976 - மான்ட்ரியல், க
னடா 19. 1980 - மாஸ்கோ, USSR 20. 1984 - லாஸ் ஏஞ்சல்ஸ், அமெரிக்கா 21. 1988 - சியோல், தென் கொரியா 22. 1992 -
பார்சிலோனா, ஸ்பெயின் 23. 1996 - அட்லாண்டா, அமெரிக்கா 24. 2000 - சிட்னி, ஆஸ்திரேலியா 25. 2004 - ஏதென்ஸ்,
கிரீஸ் 26. 2008 - பீஜிங், சீனா 27. 2012 - லண்டன், இங்கிலாந்து 28. 2016 - ரியோ, பிரேசில் 29. 2020 - டோக்கியோ, ஜ
ப்பான் 30. 2024 - பாரிஸ், பிரான்ஸ் 31. 2028 - லாஸ் ஏஞ்சல்ஸ், அமெரிக்கா Music ncs:
இன்னும் வாழ்வதில் நம்பிக்கையற்றுப்போன குடும்பத்தின் எஞ்சிய உருப்படி இவன்தான். கடுமையான எறிகணைத் தாக்குத
லொன்றில் இந்த 19வயது இளைஞனும் இவனது குடும்பத்து உறுப்பினர்கள் யாவரும் காயமடைந்து போனார்கள். இனி
வாழ்வில்லை ஒன்றாய் யாவரும் பிணமாகப் போகிறோம் என்ற நினைவில் இரத்த வெள்ளத்தில் கிடந்தார்கள். கையையிழந்த
தந்தை கையில் காயங்களுடன் அண்ணன் வயிற்றில் காயத்தோடு தங்கை கையிலும் காலிலும் காயத்தோடு தம்பி உடலில் எ
றிகணைச் சிதறல்களை ஏந்திய அம்மாவென குடும்பத்தில் சற்றுக் குறைந்த காயத்தோட ஆரோக்கியமானது இவன் ஒருவன்
மட்டும்தான்.
20வயதில் போராளியாகி 20 வயதிலே களமொன்றில் காயமடைந்து இடுப்பின் கீழ் உணர்வுகள் இழந்த போதும் , தான்
நேசித்த மண்ணுக்காகத் தனது ஆற்றல் முழுவதையும் அர்ப்பணித்து 2009 மே17, வரையும் களத்தில் கடைசி மூச்சையும் அ
ர்ப்பணிக்கும் முடிவோடு காத்திருந்தவன். நிலமைகள் நினைத்தவற்றுக்கு மாறாக தலைகீழாகி இவனதும் இவன் போன்ற ஆ
யிரமாயிரம் பேரினதும் கனவுகளில் துரோகங்கள் வென்றுவிட தோற்றுப்போனது தமிழினம். கண்ணீரோடு கடைசியாக
முள்ளிவாய்க்காலிலிருந்து இவனை இவனது தங்கைகள் காத்துக் கொண்டு போனார்கள். இவனை தடுப்பில் அடைத்தார்கள்.

Figure 1: Snapshot of the Tamil Corpus

containing a single column named `text`. To ensure a fair comparison across tokenizers, a controlled experimental setup was adopted in which the model architecture remained constant while only the tokenization strategy differed. We trained three subword tokenizers from scratch, initializing their algorithm configurations based on standard pretrained models:

WordPiece (WP): Configuration initialized from `bert-base-uncased`.

Byte-Level BPE (BBPE): Configuration initialized from `roberta-base`.

SentencePiece (SP): Configuration initialized from `albert-base-v1`.

Note: These references indicate the tokenization algorithm type only; no pretrained vocabulary or weights were transferred.

For the language model, we employed a custom lightweight Transformer architecture for all three settings with 3 encoder layers, 3 attention heads per layer, and a hidden layer size of 192 units,and the vocabulary size was set to 20,000. Furthermore, we used the following configurations for pretraining the models: masking 15% of the input tokens, a maximum sequence length of 64, a batch size of 64, and the 'AdamW' optimizer, with a learning rate of 0.0001. In addition, a Word Level tokenizer is pretrained and employed as a control baseline to facilitate comparison with subword based tokenization methods.

4.3. Finetuning for Downstream Task

For the training of Tamil news classification models, a structured procedure was followed to ensure reproducibility. The process began by setting a fixed seed value for consistency across runs. The tokenizer and model were then loaded using the pretrained weights. The tokenizer was utilized to preprocess the dataset by truncating and padding input sequences to 64 tokens.

For the downstream classification task, we employed a BERT-based transformer architecture. This choice aligns with recent comparative tokenization studies in morphologically rich languages, such as (Qarah and Alsanoosay, 2024), which standardized on `BertForSequenceClassification` to isolate tokenization variables across Arabic NLP tasks. While we acknowledge potential architecture-tokenizer affinity (e.g., WordPiece being native to BERT), we conducted preliminary validation using `AlbertForSequenceClassification` and `RobertaForSequenceClassification` models. These experiments confirmed that the relative performance ranking of tokenizers (WordPiece > SentencePiece > BBPE) remains consistent across architectures, indicating that the observed performance gaps are driven by tokenization quality rather than model compatibility alone. Consequently, we report results using the BERT architecture, which yielded the highest overall accuracy among the three classifiers tested.

For evaluation, accuracy and F1-score were computed. In addition, misclassified samples were identified and logged for qualitative analysis. A set of training arguments was configured, including 5 epochs, batch size 64, and learning rate 0.0001. Mixed precision training (fp16) was enabled. The training process was executed using the Hugging Face Trainer API. The procedure was carried out 5 times setting different seeds to obtain an average performance.

4.4. Dataset

The dataset used for the downstream task is obtained from Kaggle Vijaya Bhaskar (2020) which is a collection of Tamil news data collected from the Tamilmurasu website. When observing the count of news categories, which is in the news category column in the dataset, a high class imbalance was observed as shown in Figure 2. Since the prediction of BERT models is affected by class imbalance (Madabushi et al., 2019), the dataset needed to be balanced. To achieve this, categories whose counts were greater than 7000 were selected for

	news_category	Count
0	ஆன்மீகம்	406
1	இந்தியா	16935
2	உலகம்	7477
3	கல்வி	240
4	குற்றம்	16290
5	சினிமா(ரீல்மா)	9248
6	தமிழகம்	53333
7	தலையங்கம்	1535
8	தொழில்	68
9	மருத்துவம்	544
10	மர்மம்	66
11	மாவட்ட மசாலா	9079
12	விளையாட்டு	8230
13	வேலைவாய்ப்பு	1042
14	ஸ்டேட் எக்ஸ்பிரஸ்	2253

Figure 2: Snapshot of the count of News Categories

the final dataset. Figure 3 shows the news categories and their distribution in the balanced dataset. The dataset was split into 80% for training and 20% for testing. To maintain the same balanced distribution in the train set, the stratify option of `train_test_split` was used.

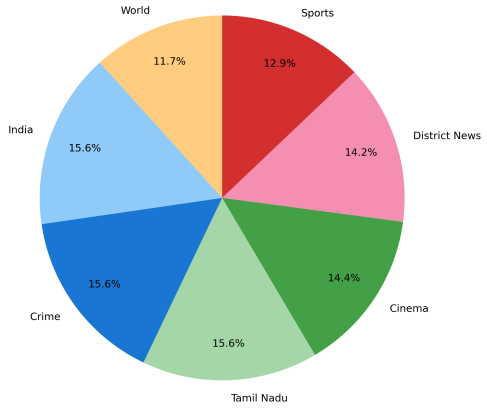

Figure 3: Piechart for distribution of News Categories after balancing

The news titles column served as the input source for both testing tokenization quality metrics and downstream classification tasks, where instances are categorized into one of several predefined categories. An example of the tokenization of a text from the news titles using the pretrained models is shown in figure 4.

4.5. Evaluation Metrics

To evaluate the performance of the tokenizers and the downstream model, we employ the following metrics.

4.5.1. Tokenization Metrics

Token Count per Sentence (TCS) This metric computes the average number of tokens produced per sentence after tokenization:

$$\text{Avg. Token Count} = \frac{1}{N}\sum_{i=1}^{N} T_i \tag{1}$$

where N is the total number of sentences and T_i denotes the number of tokens in the i-th sentence. A higher token count indicates frequent subword splitting.

Subword Fragmentation Score (SFS) This measures how often words are split into multiple subwords:

$$\text{Frag. Score} = \frac{\text{No. of Subword Tokens}}{\text{No. of Original Words}} \tag{2}$$

A higher score reflects excessive word breaking, which may negatively affect language understanding.

Out-of-Vocabulary Rate (OOVR) The OOV rate measures how frequently tokens are mapped to the unknown token (`<unk>`):

$$\text{OOV Rate} = \frac{\text{No. of OOV Tokens}}{\text{Total No. of Tokens}} \tag{3}$$

A lower OOV rate indicates better vocabulary coverage.

Compression Ratio (CR) This evaluates how efficiently the tokenizer represents the input text:

$$\text{Comp. Ratio} = \frac{\text{Original Seq. Length}}{\text{Tokenized Seq. Length}} \tag{4}$$

Here, lengths refer to word count and token count, respectively. A higher ratio implies more efficient tokenization.

4.5.2. Classification Metrics

Accuracy. Accuracy measures the proportion of correctly classified samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives.

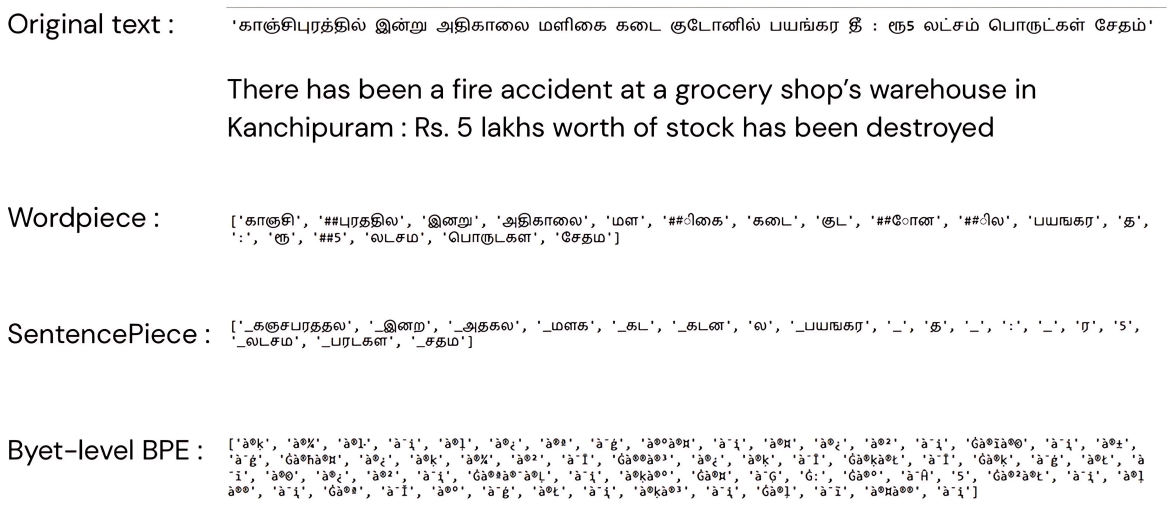

Figure 4: Snapshot of the example of the tokenization of a text

F1 Score. The F1 score is the harmonic mean of precision and recall. To save space, we define Precision (P) and Recall (R) inline: $P = \frac{TP}{TP+FP}$ and $R = \frac{TP}{TP+FN}$. The F1 score is then:

$$\text{F1} = \frac{2 \times P \times R}{P + R} \tag{6}$$

This metric is particularly useful for evaluating performance on imbalanced datasets.

5. Results and Discussion

Metric	WP	SP	BBPE
TCS	11.332	**10.353**	45.503
SFS	1.529	**1.381**	6.115
OOVR	0.060	0.059	**0.000**
CR	5.330	**5.835**	1.328

Table 1: Tokenization Quality Metrics Results

From Table 1, Our comparative analysis demonstrates clear distinctions among Byte-Level BPE (BBPE), WordPiece (WP), and SentencePiece (SP) across multiple evaluation metrics. BBPE produces substantially more tokens per sentence (45.503) relative to WordPiece (11.332) and SentencePiece (10.353), reflecting its tendency to decompose words into fine grained subword units. This behavior is corroborated by the subword fragmentation score, where BBPE registers 6.115 fragments per word, compared to 1.529 for WordPiece and 1.381 for SentencePiece. The higher fragmentation in BBPE yields longer input sequences, which can increase memory consumption and slow both training and inference.

Compression ratio and token overlap metrics reinforce these observations. BBPE attains the lowest compression ratio (1.328), confirming that its tokenized output expands significantly relative to the original text. In contrast, WordPiece (5.330) and SentencePiece (5.835) achieve markedly higher compression, indicating greater information density per token.

Taken together, these results indicate that BBPE prioritizes universal coverage at the expense of sequence length, vocabulary diversity, and computational efficiency. WordPiece and SentencePiece strike a more favorable balance for downstream classification tasks, offering compact representations, manageable vocabulary sizes, and reasonable out of vocabulary handling. For resource constrained settings or architectures sensitive to input length, the latter two methods present a more practical choice without substantially sacrificing lexical coverage.

Metrics	Word level	WP	SP	BBPE
Accuracy	72.86%	**75.55%**	74.32%	56.88%
F1 Score	72.93%	**75.58%**	74.36%	54.64%

Table 2: Train set results classification with 20000 vocabulary

Metrics	Word level	WP	SP	BBPE
Accuracy	67.76%	**72.42%**	70.73%	50.66%
F1 Score	67.51%	**72.34%**	70.55%	48.97%

Table 3: Test set results classification with 20000 vocabulary

From the Table 2 and 3 results, it can be observed that WordPiece tokenization consistently yields the highest performance across both training and test sets. It achieves 72.42% accuracy and 72.34% F1 score on the test set. SentencePiece closely follows. The Word-Level baseline shows modest performance, while Byte-Level BPE performs significantly worse, especially on the test set where it achieves only 50.66% accuracy, indicating poor generalization.

The confusion matrix shown in Figures 5, 6, and 7 illustrates how each category is predicted by the tokenizers in the test set. The description of the numbers that represent each category is Cinema: 0, Crime: 1, District News: 2, India: 3, Sports: 4, Tamil Nadu: 5, World: 6

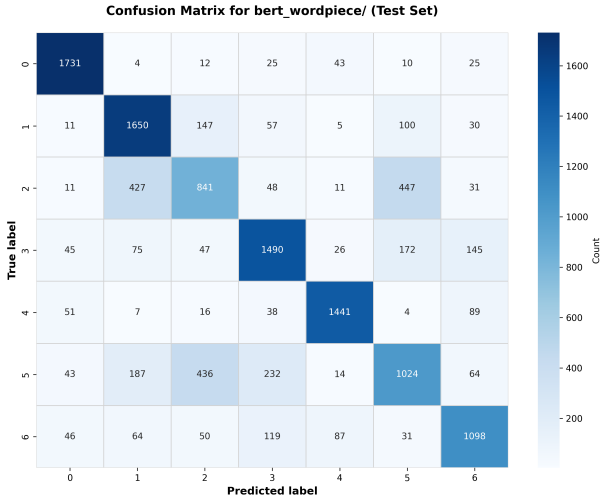

Figure 5: Confusion matrix for WordPiece

A qualitative error analysis was conducted on both misclassified and correctly classified samples to comprehensively understand the relationship between tokenization and classification performance. For misclassified samples, errors primarily stemmed from over-fragmentation and under-fragmentation in WordPiece and SentencePiece, particularly in the case of compound or morphologically rich words. Additionally, both subword methods occasionally segmented named entities (such as city or person names), which, while not inherently problematic for classification, can degrade model performance when such segments lose semantic meaning. However, to ensure these patterns are indeed correlated with classification errors rather than occurring universally, we also examined correctly classified samples. Our analysis revealed that properly classified instances generally exhibited appropriate tokenization boundaries without excessive over-fragmentation or under-fragmentation. This comparative analysis confirms that tokenization quality directly influences downstream classification accuracy. Figures 8 and 9 illustrate the over-fragmentation and under-fragmentation issues observed in misclassified samples for WordPiece and SentencePiece tokenization, respectively. In contrast, Figures 10 and 11 demonstrate proper tokenization with correct classification for WordPiece and SentencePiece, showing well-formed token boundaries that preserve semantic meaning. These patterns suggest tokenization errors can propagate downstream and affect classification accuracy, while appropriate tokenization supports reliable model performance.

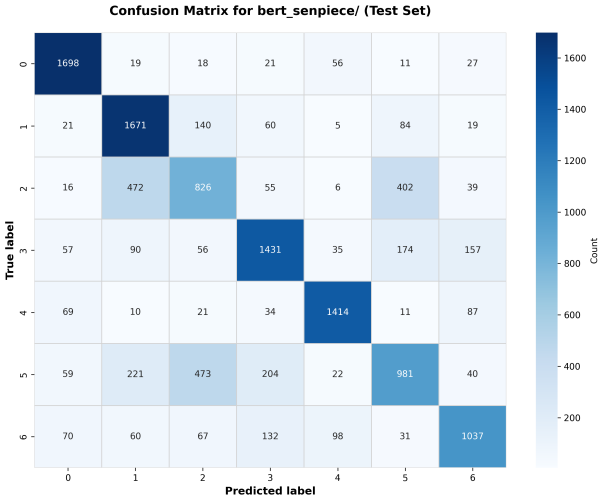

Figure 6: Confusion matrix for Sentencepiece

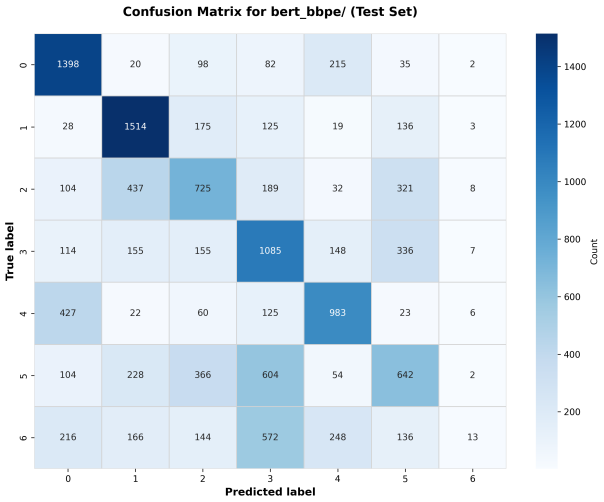

Figure 7: Confusion matrix for Byte-Level BPE

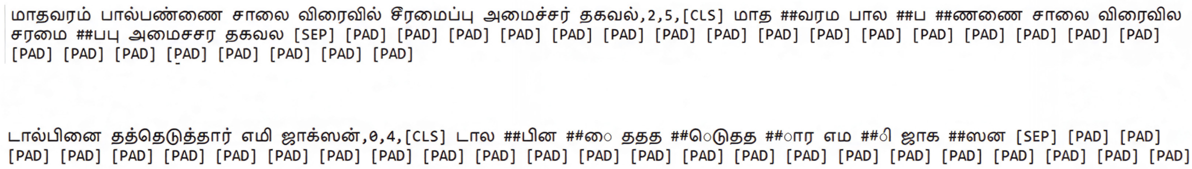

Figure 8: WordPiece Segmentation in misclassification

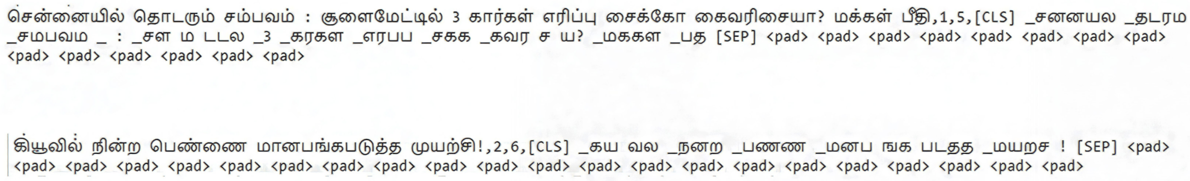

Figure 9: SentencePiece Segmentation in misclassification

In contrast, analyzing errors for BBPE proved more complex. Unlike WordPiece and SentencePiece, BBPE outputs encoded byte sequences, which are not readily interpretable in human-readable subwords. Decoding was necessary to reconstruct the original input for meaningful analysis.

To improve the accuracy of the tokenizers, particularly WordPiece and SentencePiece, which exhibited tokenization issues, they were pretrained with a larger vocabulary of 50,000. This adjustment is focused to address fragmentation problems and enhance their tokenization efficiency. Additionally, BBPE was also pretrained with a 50,000 vocabulary size to evaluate whether increasing the vocabulary would lead to any significant changes in classification performance. The results of the classification task after pretraining each tokenizer with a 50,000 vocabulary size are presented in Table 4 and 5.

Metrics	WP	SP	BBPE
Accuracy	**76.19%**	75.07%	56.30%
F1 Score	**76.23%**	75.10%	54.35%

Table 4: Train set results classification with 50000 vocabulary

Metrics	WP	SP	BBPE
Accuracy	**73.05%**	71.69%	50.24%
F1 Score	**72.96%**	71.57%	47.35%

Table 5: Test set results classification with 50000 vocabulary

The results show a modest improvement for both WordPiece and SentencePiece tokenizers when the vocabulary size is increased. WordPiece reached a peak training accuracy of 76.19% and test accuracy of 73.05%, suggesting enhanced learning and better generalization. SentencePiece also showed slight improvements in both training and test metrics. However, BBPE did not benefit from the larger vocabulary. In fact, its performance slightly declined in the training set and remained almost unchanged in the test set, with only 50.24% accuracy and 47.35% F1 score. An additional attempt was made to improve BBPE's performance by applying a pre-tokenization strategy that first split the input based on whitespace before feeding it to the BBPE model. Despite this modification, there was no significant improvement in its classification metrics.

6. Conclusion

This study presents a systematic empirical evaluation comparing tokenization strategies for Tamil language modeling using both intrinsic (tokenization quality) and extrinsic (news classification) assessments. The findings reinforce that subword-based tokenization, especially WordPiece (72.42%) and SentencePiece (70.73%), can be effective for Tamil in the Text Classification task and also maintain efficiency in tokenization. While BBPE excels in eliminating OOV words, its over-fragmentation hampers semantic modeling and Text Classification downstream performance (50.66%). The practical takeaway is that tokenizer choice significantly impacts NLP outcomes and must balance efficiency, semantic preservation, and model compatibility. By understanding the strengths and weaknesses of each tokenizer, future Tamil NLP applications can be better equipped to handle the challenges of agglutinative morphology and data scarcity.

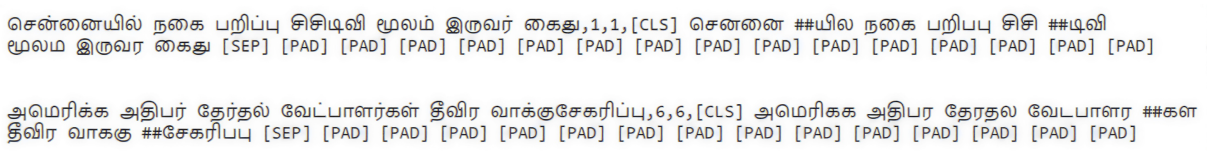

Figure 10: WordPiece Segmentation in correct classification

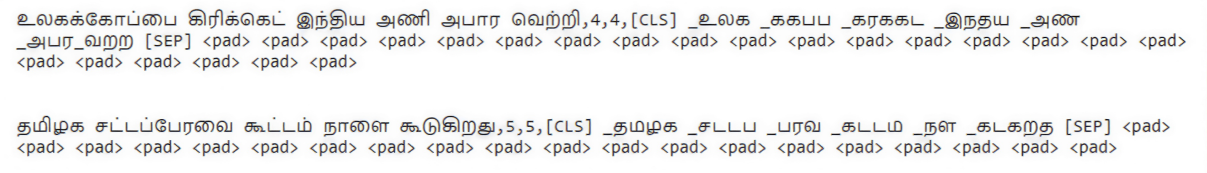

Figure 11: SentencePiece Segmentation in correct classification

7. Limitations

This study faced several limitations, primarily stemming from computational constraints, which concurrently outline the trajectory for future research. Specifically, a larger Tamil corpus for pretraining was not utilized, suggesting that future work should focus on leveraging extensive Tamil corpora to enhance representation learning. Furthermore, the maximum sequence length was restricted to 64 tokens; subsequent research could benefit from increasing this length to capture longer contextual dependencies. Due to limited GPU resources, simpler transformer architectures were employed, indicating that exploring deeper transformer models and hybrid tokenization approaches remains a key avenue for improvement. For Tamil characters requiring multiple bytes in UTF-8, byte-level approaches may introduce additional overhead. Character-level BPE should be evaluated to isolate encoding effects from tokenization algorithm effects. Finally, while the evaluation was confined to Text Classification due to the short sequence length's inability to capture longer sequences, applying tokenization analysis to diverse tasks such as sentiment analysis and Named Entity Recognition (NER) with higher sequence length like 128 would provide a more robust assessment of the model's generalizability.

8. Ethical Considerations

The Tamil Corpus and Tami News dataset is publicly available on Kaggle under standard licensing. All code and trained models is publicly available on https://github.com/Gokul200089/Tokenizers. Experiments were run on shared HPC resources; we minimized carbon footprint by using efficient lightweight architectures. Limitations regarding dialectal coverage are discussed in Section 7.

9. Bibliographical References

Ahmed Abdelali et al. 2021. Arabic pre-trained language models. In *Proceedings of the Sixth Arabic NLP Workshop*.

Muhammad Abdul-Mageed et al. 2020. Camelbert: Surprising cross-lingual transfer results. In *Proceedings of EMNLP 2020*.

E. Annamalai and R. E. Asher. 2002. *Colloquial Tamil: The Complete Course for Beginners*. Routledge.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. In *Proceedings of the Fourth Workshop on Open-Source Arabic Corpora and Processing Tools*.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of EMNLP 2020*, pages 4145–4155.

Guy Dagan et al. 2024. Tokenization from a compression and semantic trade-off perspective. *Transactions of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2020. Optimizing word segmentation for downstream task. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1341–1351, Online. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 66–75.

Taku Kudo and John Richardson. 2018a. Sentencepiece: A simple and language independent subword tokenizer. In *Proceedings of EMNLP 2018: System Demonstrations*, pages 66–71.

Taku Kudo and John Richardson. 2018b. 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: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, et al. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, et al. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Harish Tayyar Madabushi, Elena Kochkina, and Mathieu Castelle. 2019. Cost-sensitive bert for generalisable sentence classification on imbalanced data. In *Proceedings of the Second Workshop on NLP for Internet Freedom*, pages 125–134.

Kyubyong Park, Joohong Lee, Seongbo Jang, and Dawoon Jung. 2020. An empirical study of tokenization strategies for various korean nlp tasks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 133–142, Suzhou, China. Association for Computational Linguistics.

Muhammad Qarah and Ibrahim Alsanoosay. 2024. A comparative study on tokenizers for arabic nlp tasks. In *Proceedings of the Conference on Arabic Natural Language Processing*.

Alec Radford, Jeffrey Wu, Rewon Child, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Technical Report*.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. 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 3118–3135, Online. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulic, et al. 2020. How good is your tokenizer? *arXiv preprint arXiv:2012.15613*.

K. Sarveswaran, G. Dias, and M. Butt. 2021. Thamizhi morph: A morphological parser for the tamil language. *Machine Translation*, 35(1):37–70.

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)*, pages 1715–1725.

Harshit Singh and Rajiv Ratn Shah. 2022. Tamilnlp: Low resource language processing.

M. Thangarasu and R. Manavalan. 2013. Stemmers for tamil language: Performance analysis. *International Journal of Computer Trends and Technology*, 4(8):2582–2584.

Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

K. T. Veeramanju. 2023. A systematic review on news headlines categorization in malayalam. *Scope*, 13.

Jia Wei, Qiang Liu, Yiqun Guo, and Xia Jiang. 2021. Training multilingual pre-trained language models with byte-level subwords. *arXiv preprint arXiv:2101.09469*.

Thomas Wolf, Lysandre Debut, Victor Sanh, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*, pages 38–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

10. Language Resource References

Praveen Govi. 2020. *Tamil Language Corpus for NLP*. Kaggle Datasets. Kaggle. A large collection of Tamil text corpora for natural language processing research.

Vijaya Bhaskar. 2020. *Tamil News Classification Dataset (Tamil Murasu)*. Kaggle Datasets. Kaggle. News articles collected from the Tamil Murasu newspaper and released for text classification research.