Classification of Buddhist Verses: The Efficacy and Limitations of Transformer-Based Models

Nikita Neveditsin[1], Ambuja Salgaonkar[2], Pawan Lingras[1], Vijay Mago[3]

[1]Saint Mary's University, Halifax, Canada
[2]University of Mumbai, Mumbai, India
[3]York University, Toronto, Canada

Abstract

This study assesses the ability of machine learning to classify verses from Buddhist texts into two categories: Therigatha and Theragatha, attributed to female and male authors, respectively. It highlights the difficulties in data preprocessing and the use of Transformer-based models on Devanagari script due to limited vocabulary, demonstrating that simple statistical models can be equally effective. The research suggests areas for future exploration, provides the dataset for further study, and acknowledges existing limitations and challenges.

1 Introduction

The term "gāthā" (gatha) denotes a poetic meter primarily employed in legends and folklore, yet it is notably absent from the Vedas (Mukherjee, 1998). Gathas are popular in Maharashtra, India, where locals are familiar with the gathas of Tukaram (Tukaram, 2014). However, the earliest known reference to gathas appears in the Avesta, a Zoroastrian scripture compiled during the Sasanian Empire (224-651 BCE) (Hintze, 2002). The languages in which these ancient gathas were written have since become extinct. Consequently, interpreting them is challenging and necessitates reliance on extant languages that exhibit similar, yet distinctly different, structures.

This study examines two collections from the Buddhist canonical literature: *Theragathapali* and *Therigathapali*, which are, respectively, the line-wise utterances attributed to male and female saints. This literature is written in Pali, a language believed to be a mixture of Prakrit languages, closely related to the vernacular of the common people during the time of Siddhartha Gautama Buddha (circa 600 BCE).

The authorship of some gathas is debatable. Kumara (2016) observes that in Pali literature, authorship details are occasionally provided at the beginning or end of the texts. However, not all authors considered it essential to include such information. In examining the authorship of the Therigatha, Findly (1999) suggests that the authorship of some verses may be doubtful, indicating that while some verses are traditionally attributed to the female saints themselves, others may have been composed or recited by different individuals, including the possibility of later attribution by compilers. This uncertainty in authorship challenges the straightforward attribution of these texts to the female saints they are associated with.

Nevertheless, studies demonstrate that the Theri gathas differ from the Thera gathas. Blackstone (2013) argues that the Theri gathas focus more on themes of overcoming suffering, societal constraints, and personal liberation. A study by Marques et al. (2021) confirms the uniqueness of topics in Therigatha.

Typically, a gatha is a two-line verse, although variations include verses comprising three or four lines. Figure 1 provides a sample two-line gatha in Devanagari script.

यो पुब्बे करणीयानि, पच्छा सो कातुमिच्छति।
सुखा सो धंसते ठाना, पच्छा च मनुतप्पति॥

Figure 1: Sample Gatha in Devanagari Script.

Banerjee (2017) suggests that translations of gathas influences the perception of these ancient texts. For example, the gatha from Figure 1 is translated by Bhikkhu (1998) as "Whoever wants to do later what he should have done first, falls away from the easeful state and later burns with remorse", while one of the contributors of this study translates the second line as "He destroys pleasure producing points and regrets later".

The abundance of Transformer-based models (Vaswani et al., 2017) and their proficiency across various domains (Fisher et al., 2023; Phatak et al.,

Proceedings of the 4th International Conference on Natural Language Processing for Digital Humanities, pages 377–385
November 16, 2024.

2024; Neveditsin et al., 2024), particularly in classification tasks (Munikar et al., 2019; Kheiri and Karimi, 2023; Hartmann et al., 2023; Zielinski et al., 2023; Zaczynska et al., 2024), inspired us to conduct a study on their performance in classifying verses from low-resource Pali texts. While we acknowledge the debates around the authorship of some Therigatha verses, we deliberately avoid this discussion in our study due to the lack of definitive evidence regarding authorship. Consequently, we treat Therigatha verses as authored by female authors and Theragatha verses as authored by male authors.

The goal of this study is to determine whether Transformer-based models can outperform traditional machine learning models in the binary classification of the verses. We hypothesize that Transformer-based models, even when pretrained on languages other than Pali, can still identify patterns specific to each class. Additionally, we aim to assess the performance difference of these models when using Devanagari script versus Roman script. Through this investigation, we aim to highlight the challenges associated with this task and suggest directions for future research.

2 Related Work

Research on poetry classification in the Pali language using machine learning is scarce, however, insights can be drawn from related areas, including poetry classification in other languages, text classification in low-resource settings, and computational analysis of Pali texts.

One of the earliest studies in poetry classification is by Kao and Jurafsky (2012), who use logistic regression to examine stylistic and content features that distinguish professional poets from amateurs. The authors extract features related to diction, sound devices, affect, and imagery to identify elements contributing to poetic sophistication. Similarly, Pal and Patel (2020) classify Hindi poems using machine learning, providing insights into poetry classification in an Indo-Aryan language closely related to Pali. The authors employ classical models, such as Naïve Bayes, Random Forest, and SVM, achieving a maximum accuracy of 64% with Naïve Bayes, highlighting the challenges of poetry classification due to the morphological richness and varied sentence structures.

In the context of text classification for low-resource languages, recent research suggests that cross-lingual models, such as XLM (Lample and Conneau, 2019), may sometimes offer performance gains compared to classic machine learning models like SVM or Naïve Bayes. For instance, Li et al. (2020) introduce a model called AgglutiFiT, fine-tuned from a cross-lingual pre-trained model (XLM-R), which significantly outperforms strong baselines in terms of accuracy.

Additionally, Alekseev et al. (2024) benchmark multilabel topic classification in the Kyrgyz language, evaluating several baseline models, including classical approaches and neural models like XLM-RoBERTa. Their findings indicate that the multilingual model XLM-RoBERTa outperforms classical models in terms of F1 score. However, transformer-based models do not always surpass traditional machine learning models for low-resource languages. For example, Lalthangmawii and Singh (2023) found that the SVM model achieved the highest accuracy (75%) on a sentiment classification task for the Mizo language, performing similarly to the XLM-RoBERTa model using a transfer learning approach.

Another method for handling low-resource languages is leveraging machine translation. Recent work by Kumar et al. (2024) provides valuable insights into sentiment classification for low-resource Indian languages using machine-translated datasets. The results highlight the potential of datasets translated with tools like Google Translate and indicate that models such as LSTM can effectively preserve sentiment by accounting for sequential patterns.

Focusing specifically on Pali texts, Zigmond (2021) conduct a computational analysis of the Pali Canon. The author uses computational text mining to examine various volumes of the Canon, extracting linguistic and thematic insights. By employing techniques such as k-means clustering and Principal Component Analysis, they reveal differences between older texts (Vinaya and Suttas) and later ones (Abhidhamma). The research also underscores the complexity of Pali language processing, including multiple word declensions, elisions, and compound formations.

3 Dataset

The dataset utilized in this study comprises the Thera and Theri gatha texts from the Khuddakanikaya volume of the Sutta-pitaka, which is the third part of the Buddhist canonical literature, Tip-

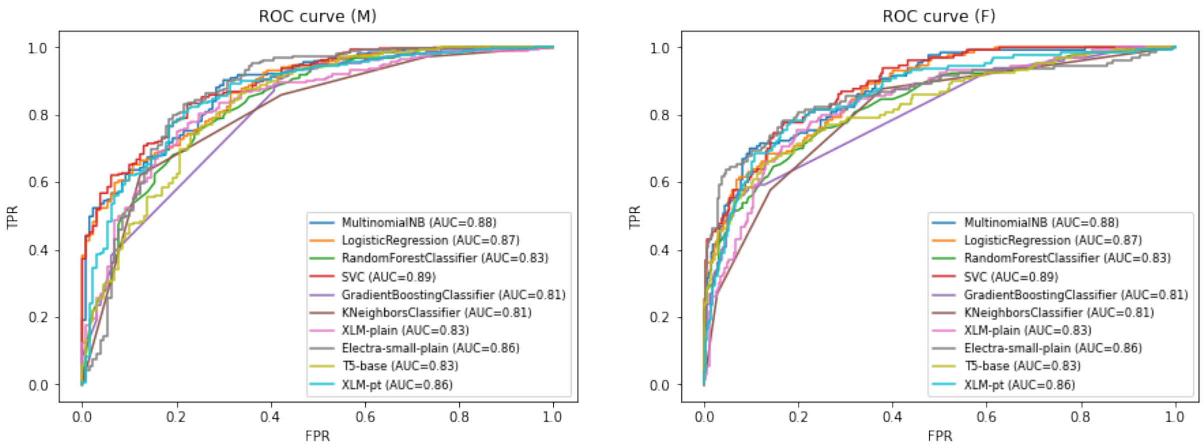

Figure 2: ROC curves for Roman script classification. Left: results for ‘M’ class (Theragatha); right: ‘F’ class (Therigatha). Multiple models are compared, with AUC scores indicating performance.

itaka[1]. Each gatha is categorized into chapters based on the number of verses attributed to each author: single verses are compiled in the chapter named *Ekaka-nipaat*, meaning ‘collection of ones’, while chapters such as *Dukanipaat*—‘collection of twos’—contain texts with two verses from a single author, and so forth. The Theragatha consists of 1,288 verses spread across 21 chapters, whereas the Therigatha contains 524 verses distributed over 16 chapters, with all verses sequentially labeled within their respective compendiums.

To study the potential impact of script on the training of the classifier, both the Devanagari and Roman versions were used. The manual preprocessing involved several steps:

1. Punctuation Handling: We agreed on approaches to interpret punctuation marks, considering variations in their usage across different scripts.
2. Text Completion: This addresses instances of "*peyaala*" (or "*pe*"), which indicate a repetition of words or lines from previous parts of the text. Due to the lack of suitable computational linguistic tools for this task, matching the context of *peyaala* to find the appropriate text from earlier sections was conducted manually.
3. Word Separation: Ancient Indian languages feature notable word compounding and clubbing. Unlike Sanskrit, where the rules for word combination are relatively rigid, Pali allows more flexibility. This necessitates greater care in separating compounded words into their individual components. Due to the challenges in separating these combined words, we decided to work with the combined forms as they appear in the text.

After the manual preprocessing of the text, we encountered discrepancies in the counts of distinct words when tokenizing the verses by spaces. Assuming a one-to-one correspondence between tokens in the Devanagari and Roman scripts, a dictionary-based test was applied to identify these discrepancies. The test revealed several transliteration nuances. For instance, some symbols such as नो and खो in Devanagari are represented by two UTF-8 code points, which leads to confusion with symbols न and ख, respectively. Another challenge was caused by complex compounding rules; for example, space-based tokenization ambiguously mapped the symbol मुनि to either ‘*muni*’ or ‘*munin*’, depending on the neighboring tokens (a one-to-many case). Similarly, both symbols न्ति and ति map to ‘*ti*’ in the Romanized script (a many-to-one case). These cases demonstrate that space-based tokenization may not adequately capture the nuances of these complex verses. For this study, we decided to exclude three nuanced verses from the Theri gathas and sixteen nuanced verses from the Thera gathas where we were unable to easily resolve the inconsistencies. This resulted in 1793 verses in our dataset[2]. Table 1 presents the statistics on word distribution among the scripts.

4 Experiments and Results

The overall task can be defined as a binary classification problem with two categories: ‘M’ for Theragathas and ‘F’ for Therigathas. The dataset, divided

[1] Digital version available here: https://tipitaka.org/

[2] https://github.com/neveditsin/pali

Statistic	Dev.	Rom.
Total Distinct Words	8787	8789
Female Unique Words	3145	3143
Male Unique Words	6548	6547
Only Female Words	2239	2242
Only Male Words	5642	5646
Common Words	906	901

Table 1: Word Distribution in Devanagari (Dev.) and Roman (Rom.) Texts.

by script type into Devanagari and Roman subsets, was split into training (75%) and test (25%) sets. Considering the dataset's imbalance, we report key metrics such as ROC-AUC, Matthews Correlation Coefficient (MCC), as well as precision, recall, F1-scores, and average precision (AP) for both classes. We deliberately avoided sampling to address the imbalance due to the dataset's small size. However, by providing a comprehensive set of metrics, we aim to give a detailed comparison of the models' performance across different aspects.

First, we applied traditional machine learning models: Multinomial Naïve Bayes, Logistic Regression, Random Forest Classifier (RFC), Support Vector Classifier (SVC), Gradient Boost Classifier (GBC), and K-Nearest Neighbors Classifier (KNN), on the Roman script to classify gathas. Space tokenization and a TF-IDF matrix were used for all models except for the Multinomial Naïve Bayes, which served as a baseline model using simple count vectorization. The Multinomial Naïve Bayes assumes conditional independence of tokens and positional independence of features. Naïve Bayes can be optimal under certain circumstances, such as when the conditional independence assumption holds (Zhang, 2004). To assess whether transformer-based models could improve specific aspects of classification, such as precision and recall, we experimented with fine-tuning the following models: XLM (Lample and Conneau, 2019), XLM pre-trained additionally on our training corpus, T5-base (Raffel et al., 2023), and Electra-small (Clark et al., 2020). Figure 2 presents the classification results for the Roman script.

Similar experiments with the Devanagari script revealed that while transformer-based models underperformed relative to their counterparts in Roman script, the performance disparities among traditional models were minimal, as depicted in Figure 3. Additionally, our trials with a byte-level T5 (Xue et al., 2022) model yielded substantially lower performance (AUC 0.58 for Devanagari), which we attribute to its inability to effectively handle script-specific complexities, leading to its exclusion from our study.

When investigating why transformer-based models exhibit inferior performance compared to classic machine learning algorithms, we analyzed the number of tokens generated by tokenizers for both Devanagari and Roman scripts in the test subsets. Table 2 presents the counts of unique tokens from the tokenizers applied to the test set. Our analysis revealed a strong correlation between the number of tokens and classification outcomes. This suggests that the underperformance of transformer-based models on the Devanagari script is attributed to significant information loss during tokenization with certain tokenizers.

	ByT5	OpenHathi	T5	XLM	Electra
Devanagari Tokens	54	1200	6	1208	60
Roman Tokens	44	-	748	1909	1313

Table 2: Unique Tokens in Test Subsets by Model

To address this issue, we opted to fine-tune OpenHathi-7B (Sarvam, 2024), a model based one Llama-2 (Hugo Touvron, 2023), specifically developed for Indo-Aryan languages. We utilized Low-Rank Adaptation (LoRA) (Hu et al., 2021) to adjust the model's parameters, using the last token for classification purposes. Notably, even after fine-tuning, the OpenHathi model did not outperform the simpler XLM model.

Table 3 provides detailed classification results for the best performing models compared to Multinomial Naïve Bayes. Notably, OpenHathi exhibited the highest recall for the minority class among the evaluated models. However, a paired bootstrap test (Berg-Kirkpatrick et al., 2012) with 10^5 iterations indicated that this increase in recall is not statistically significant ($p = 0.08$).

Script	Class	Precision	Recall	F1	AP	AUC	MCC
Multinomial Naïve Bayes							
Devanagari	M	0.85	0.92	0.88	0.94	0.88	0.56
	F	0.75	0.60	0.67	0.78		
Roman	M	0.85	0.92	0.88	0.94	0.88	0.56
	F	0.75	0.60	0.67	0.78		
SVC							
Devanagari	M	0.86	0.88	0.87	0.95	0.89	0.53
	F	0.68	0.64	0.66	0.80		
Roman	M	0.86	0.88	0.87	0.95	0.89	0.53
	F	0.68	0.64	0.66	0.80		
XLM							
Devanagari	M	0.80	0.91	0.85	0.89	0.78	0.42
	F	0.68	0.45	0.54	0.64		
Roman	M	0.79	0.93	0.86	0.92	0.83	0.40
	F	0.71	0.39	0.50	0.67		
XLM with Pre-Training							
Devanagari	M	0.77	0.99	0.87	0.93	0.86	0.45
	F	0.95	0.28	0.44	0.75		
Roman	M	0.78	0.97	0.87	0.93	0.86	0.44
	F	0.83	0.35	0.49	0.75		
OpenHathi-7B							
Devanagari	M	0.85	0.68	0.76	0.86	0.76	0.36
	F	0.47	0.70	0.57	0.63		

Table 3: Detailed Results for Selected Models.

Appendix A lists the hyperparameters used for

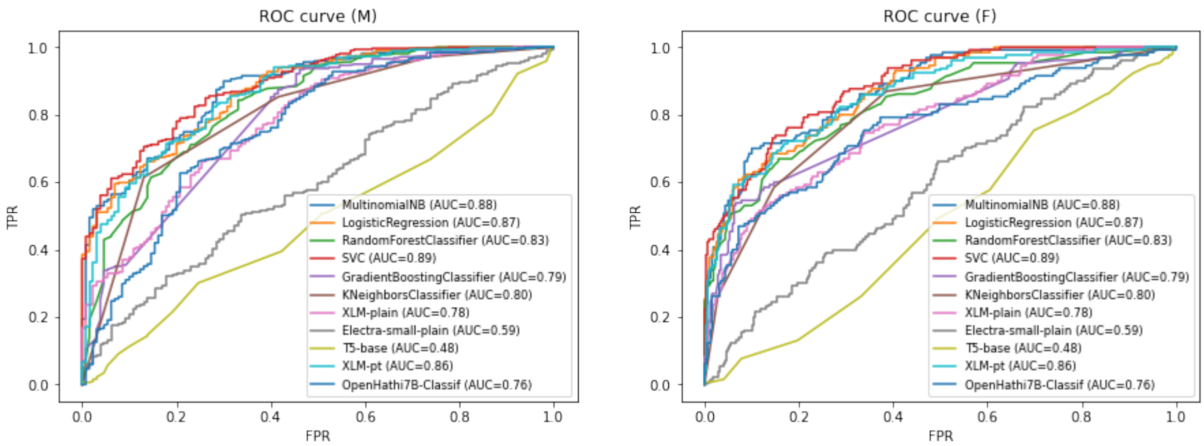

Figure 3: ROC curves for Devanagari script classification. Left: results for 'M' class (Theragatha); right: 'F' class (Therigatha). Multiple models are compared, with AUC scores indicating performance.

model training. Hyperparameters not listed are set to their default values in the scikit-learn library for classic machine learning models and in the Hugging Face Transformers library for transformer-based models.

Our attempt to employ the SHAP framework (Lundberg and Lee, 2017) on the best-performing models to explain their discrimination decisions did not reveal any specific features that contribute significantly to either of the classes.

5 Discussion and Further Research

The study highlights persistent challenges in using original, non-Romanized scripts with modern transformer-based models for classification tasks, primarily due to inadequate token coverage in the models' vocabularies. Previous studies, such as the one by Maronikolakis et al. (2021), showed that the compatibility of tokenizations is crucial in multilingual language models, discussing the importance of vocabulary size. More recently, Ali et al. (2024) confirmed that the choice of tokenizer significantly impacts a model's downstream performance. They suggest that tokenizers not tailored to handle a variety of scripts can lead to inefficient tokenization, directly affecting model performance, and that larger vocabulary sizes are required for multilingual tokenizers compared to those designed for English only.

Although Romanized versions of the scripts enabled the use of a broader range of models, these models still did not surpass the performance of traditional machine learning algorithms. This outcome suggests that the employed models failed to identify any class-specific patterns within the dataset, likely because these models lacked sufficiently relevant data during their pretraining stages. Notably, additional pre-training of the XLM model improved the AUC on the classification task, and a paired bootstrap test with 10^5 iterations confirmed the statistical significance of this improvement ($p < 0.05$).

Extended research is necessary for the authorship attribution task. Our next step is to identify Therigathas that are consistently misclassified by the majority of models and perform a detailed analysis of these cases. This includes annotating and analyzing specific gathas whose authorship is disputed by scholars. Statistical sampling to identify whether the differences between the Theri and Thera gathas are significant may help reveal if there are substantial distinctions between the two classes of gathas from a machine learning perspective. Additionally, compiling an extensive Pali corpus to pre-train a transformer model would enable us to experiment with its discriminatory abilities and its capability to generate novel gathas.

6 Limitations

First, our dataset is small and imbalanced, with only slightly over 10% of words shared between the Thera and Theri gathas. This low overlap might explain why classical machine learning algorithms were able to effectively discriminate between the classes, primarily by relying on words unique to specific classes.

The second limitation pertains to the existing transformer models, which often lack the comprehensive vocabulary necessary for thorough evaluation.

References

Anton Alekseev, Sergey Nikolenko, and Gulnara Kabaeva. 2024. Benchmarking multilabel topic classification in the kyrgyz language. In *Analysis of Images, Social Networks and Texts: 11th International Conference, AIST 2023, Yerevan, Armenia, September 28–30, 2023, Revised Selected Papers*, page 21–35, Berlin, Heidelberg. Springer-Verlag.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Schulze Buschhoff, Charvi Jain, Alexander Arno Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. 2024. Tokenizer choice for llm training: Negligible or crucial? *Preprint*, arXiv:2310.08754.

Supriya Banerjee. 2017. Ambapali's verse in therigatha: Trajectories and transformations. *Translation Today*, 11:15.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Thanissaro Bhikkhu. 1998. Harita (2) (thag 3.15). Access to Insight (BCBS Edition), 4 August 2010.

Kathryn R Blackstone. 2013. *Women in the Footsteps of the Buddha: Struggle for Liberation in the Therigatha*. Routledge.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *Preprint*, arXiv:2003.10555.

Ellison Banks Findly. 1999. Women and the "arahant" issue in early pali literature. *Journal of Feminist Studies in Religion*, 15(1):57–76.

Andrew Fisher, Matthew Maclaren Young, Doris Payer, Karen Pacheco, Chad Dubeau, and Vijay Mago. 2023. Automating detection of drug-related harms on social media: machine learning framework. *Journal of medical internet research*, 25:e43630.

Jochen Hartmann, Mark Heitmann, Christian Siebert, and Christina Schamp. 2023. More than a feeling: Accuracy and application of sentiment analysis. *International Journal of Research in Marketing*, 40(1):75–87.

Almut Hintze. 2002. On the literary structure of the older avesta. *Bulletin of the School of Oriental and African Studies*, 65(1):31–51.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Louis Martin et al. Hugo Touvron. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Justine Kao and Dan Jurafsky. 2012. A computational analysis of style, affect, and imagery in contemporary poetry. In *Proceedings of the NAACL-HLT 2012 workshop on computational linguistics for literature*, pages 8–17.

Kiana Kheiri and Hamid Karimi. 2023. Sentimentgpt: Exploiting gpt for advanced sentiment analysis and its departure from current machine learning. *arXiv preprint arXiv:2307.10234*.

Saurabh Kumar, Ranbir Sanasam, and Sukumar Nandi. 2024. IndiSentiment140: Sentiment analysis dataset for Indian languages with emphasis on low-resource languages using machine translation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7689–7698, Mexico City, Mexico. Association for Computational Linguistics.

S Vijitha Kumara. 2016. A chronological approach to the pali commentaries: with reference to the madhuratthavilāsinī. *Sri Lanka International Journal of Buddhist Studies (SIJBS)*, 4:32–54.

Mercy Lalthangmawii and Thoudam Doren Singh. 2023. Sentiment analysis for the mizo language: A comparative study of classical machine learning and transfer learning approaches. In *Proceedings of the 20th International Conference on Natural Language Processing (ICON)*, pages 308–317, Goa University, Goa, India. NLP Association of India (NLPAI).

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Preprint*, arXiv:1901.07291.

Xiuhong Li, Zhe Li, Jiabao Sheng, and Wushour Slamu. 2020. Low-resource text classification via cross-lingual language model fine-tuning. In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 994–1005, Haikou, China. Chinese Information Processing Society of China.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Antonis Maronikolakis, Philipp Dufter, and Hinrich Schütze. 2021. Wine is not v i n. on the compatibility of tokenizations across languages. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2382–2399, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tattiane Yu Borges Marques et al. 2021. Therīgāthā, a primeira literatura feminina no budismo: possibilidades de diálogo com o erotismo na teopoética. *Mandrágora*, 27(1):31–52.

Sujit Mukherjee. 1998. *A dictionary of Indian literature: beginnings-1850*, volume 1. Orient Blackswan.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5.

Nikita Neveditsin, Pawan Lingras, and Vijay Mago. 2024. Clinical insights: A comprehensive review of language models in medicine. *arXiv preprint arXiv:2408.11735*.

Kaushika Pal and Biraj. V. Patel. 2020. Automatic multiclass document classification of hindi poems using machine learning techniques. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–5.

Atharva Phatak, Vijay K Mago, Ameeta Agrawal, Aravind Inbasekaran, and Philippe J Giabbanelli. 2024. Narrating causal graphs with large language models. *arXiv preprint arXiv:2403.07118*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Sarvam. 2024. Announcing openhathi series. `https://www.sarvam.ai/blog/announcing-openhathi-series`. Accessed: 2024-07-26.

Sant Tukaram. 2014. *Tukaram Gatha: Enhanced by Rigved*, volume 1. Rigved Shenai.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Preprint*, arXiv:2105.13626.

Karolina Zaczynska, Peter Bourgonje, and Manfred Stede. 2024. How diplomats dispute: The un security council conflict corpus. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8173–8183.

H Zhang. 2004. The optimality of naive bayes. In *Proceedings of the the 17th International FLAIRS conference (FLAIRS2004)*, pages 562–567.

Andrea Zielinski, Calvin Spolwind, Henning Kroll, and Anna Grimm. 2023. A dataset for explainable sentiment analysis in the german automotive industry. In *Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis*, pages 138–148.

Dan Zigmond. 2021. Toward a computational analysis of the pali canon. *Journal of the Oxford Centre for Buddhist Studies*, 20.

A Training hyperparameters

A.1 Hyperparameters for Devanagari Script

Model	Hyperparameters	Values
Classic Machine Learning Models		
MultinomialNB	Vectorizer: CountVectorizer	binary = False tokenizer = lambda x: x.split() token_pattern = None
LogisticRegression	random_state	0
RandomForestClassifier	random_state	0
SVC (Support Vector Classifier)	probability random_state	True 0
GradientBoostingClassifier	random_state	0
KNeighborsClassifier	n_neighbors	3
TfidfVectorizer Parameters		
All models using TfidfVectorizer	use_idf binary tokenizer token_pattern	True False lambda x: x.split() None
Transformer-Based Models		
XLM-Roberta (plain and fine-tuned)	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps learning_rate warmup_steps weight_decay seed	10 16 steps 100 100 2e-5 500 0.01 0
Electra	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps seed	20 16 steps 100 100 0
T5 (T5-base)	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps learning_rate warmup_steps weight_decay seed	10 16 steps 50 50 2e-5 500 0.01 0
byT5 (byT5-base)	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps learning_rate warmup_steps weight_decay seed	5 8 steps 50 10 2e-5 50 0.01 0
OpenHathi (QLoRA, Sequence Classification)	lora_r lora_alpha lora_dropout bias max_length per_device_train_batch_size gradient_accumulation_steps warmup_steps max_steps learning_rate fp16	128 256 0.1 none 512 8 4 100 2000 4e-5 True

A.2 Hyperparameters for Roman Script

Model	Hyperparameters	Values
Classic Machine Learning Models		
MultinomialNB	Vectorizer: CountVectorizer	binary = False tokenizer = lambda x: x.split() token_pattern = None
LogisticRegression	random_state	0
RandomForestClassifier	random_state	0
SVC (Support Vector Classifier)	probability random_state	True 0
GradientBoostingClassifier	random_state	0
KNeighborsClassifier	n_neighbors	3
TfidfVectorizer Parameters (used in some classic models)		
All models using TfidfVectorizer	use_idf binary tokenizer token_pattern	True False lambda x: x.split() None
Transformer-Based Models		
XLM-Roberta (plain and fine-tuned)	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps learning_rate warmup_steps weight_decay seed	10 16 steps 100 100 2e-5 500 0.01 0
Electra	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps seed	20 16 steps 100 100 0
T5 (T5-base)	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps learning_rate warmup_steps weight_decay seed	10 16 steps 50 50 2e-5 500 0.01 0
byT5	num_train_epochs per_device_train_batch_size evaluation_strategy save_steps logging_steps learning_rate warmup_steps weight_decay seed	5 8 steps 50 10 2e-5 50 0.01 0