Context-Aware Homograph Disambiguation in Bangla: A Multi-Model Machine Learning Approach

Sayma Sultana Chowdhury*[1], Md Kamruzzaman Kamrul[2], Eashraque Jahan Easha[3]

[1]Shahjalal University of Science and Technology, Sylhet 3114, Bangladesh
[2]Georgia State University, Atlanta, GA, USA
[3]University of Denver, Denver, CO, USA

*Corresponding Author: Sayma Sultana Chowdhury (sayma-iict@sust.edu)

Abstract

This article emphasizes the challenge of Bangla homograph disambiguation in Text-to-Speech (TTS) synthesis, where identically spelled words (homographs) have different meanings or pronunciations based on context. We propose a machine learning-based solution integrating five approaches: Naive Bayes, Support Vector Machine (SVM), Logistic Regression, Random Forest, and XGBoost. A strict hyperparameter tuning process and rigorous feature engineering were applied to evaluate these models' effectiveness in correctly disambiguating homographs. The study focused on five homograph pairs-Dak_Dako (ডাক/ডাকো), Bol_Bolo (বল/বলো), Kal_Kalo (কাল/কালো), Komol Komlo (কমল/কমলো), and Mot_Moto (মত/মতো)-resulting in significant improvements in TTS accuracy and reduced pronunciation errors. Our findings highlight that contextual n-grams are crucial for accurate disambiguation, with SVM emerging as the top-performing model, achieving the highest accuracy across all datasets, with values ranging from 0.9215 (Komol_Komlo) to 0.9752 (Kal Kalo). The results suggest a robust framework for homograph disambiguation in resource-constrained languages like Bangla.

Key words: Homograph Disambiguation, Bangla TTS, Machine Learning, Naive Bayes, SVM, Random Forest

1. Introduction

Bangla is a linguistically enriched language spoken by over 230 million people around the world. However, it is a less explored language in the advanced research area of Natural Language Processing. One major challenge with NLP in Bangla is homograph disambiguation-words written identically but are pronounced and signify differently according to where they are used. These are big stumbling blocks to speech synthesizers, TTS, ASR, and machine translators.

Though significant amounts of work have been done on the disambiguation of homographs in languages like English, the progress has been poor in Bangla, which is a resource-poor language having rich syntax and morphological variations. Homographs like Dak_Dako (ডাক/ডাকো) or Kal_Kalo (কাল/কালো) possess a highly context-dependent nature and hence are very tricky to disambiguate. The usual rule-based methodology has proved inefficient in handling the intricacies of Bangla syntax; hence, the consideration of machine learning techniques is being considered.

This study used a range of machine learning models, including Naive Bayes, Support Vector Machine (SVM), Logistic Regression, Random Forest, and XGBoost, on five common Bangla homograph pairs: Dak_Dako (ডাক/ডাকো), Bol_Bolo (বল/বলো), Kal_Kalo (কাল/কালো), Komol_Komlo (কমল/কমলো), and Mot_Moto (মত/মতো). This work points out the capability of machine learning, through thorough hyperparameter tuning and feature engineering, to outperform state-of-the-art methodologies and thus obtain a robust solution to Bangla homograph disambiguation. The novelty of this work is in the exhaustive model comparisons done, complemented with the work on n-grams and contextual embeddings to enhance performance upon low-resource languages like Bangla.

The result of this work contributes to NLP in Bangla and sets a premise for possible future works on homograph disambiguation in other low-resource languages. The insights gained here could also improve practical applications such as speech recognition and translation systems, making them more effective for Bangla-speaking populations.

II. Literature Review

Homograph disambiguation-classification of identically

https://doi.org/10.63512/sjst.2024.1005
Received 24 October 2024, Accepted 26 February 2025.

spelled words with different meanings and pronunciation-is one of the most fundamental challenges in NLP, which becomes further challenging when the research language is morphologically rich-like that of Bangla-where correct interpretation of the contexts is imperative in applications from TTS systems and speech synthesis to machine translation. With the use of machine learning techniques, whereas earlier studies focused on the English language, the concentration has now shifted to other languages also-Manning et al. (2008), Yarowsky (1994). This paper reviews state-of-the-art developments of ML-based models designed for Bangla homograph disambiguation.

Among these, some popular models for homograph disambiguation are Naive Bayes, SVM, Random Forest, and XGBoost. Although Naive Bayes assumes feature independence, it is still efficient for text classification tasks (Zhang, 2004). On the other hand, more advanced models, such as Support Vector Machines and ensemble methods, outperform it through knowledge of sophisticated contextual patterns (Vapnik, 1995). Contextual embeddings using representations like Word2Vec and GloVe representations improve its effectiveness for languages like Bangla (Mikolov et al., 2013; Pennington et al., 2014; Hasan et al., 2017).

SVM is effective in homograph disambiguation, especially with optimized kernel functions (Joachims, 1998). It has also succeeded in part-of-speech tagging and named entity recognition (Collobert et al., 2011). Advanced tuning methods, like grid and random search, further enhance SVM's performance in tasks with contextual ambiguity, such as Bangla (Zhang et al., 2019; Rahman et al., 2021).

Random Forest, an ensemble method, enhances accuracy by merging numerous decision trees, skillfully handling noisy data (Breiman, 2001). It has demonstrated potential in Bangla NLP tasks, such as text classification and sentiment analysis (Chowdhury et al., 2018). To achieve optimal performance, meticulous tuning of crucial parameters like tree depth and sample splits is vital (Yang et al., 2021).

XGBoost, known for handling sparse data and iterative boosting, performs well in NLP tasks, including Bangla homograph disambiguation. Effective feature extraction using TF-IDF, and n-grams boosts its accuracy (Chen & Guestrin, 2016; Al Mumin et al., 2014). Studies highlight the importance of tuning parameters such as learning rates and boosting rounds for superior results (Liu et al., 2020).

Bangla homograph disambiguation remains underexplored. Traditional rule-based approaches struggle with Bangla's complex morphology (Islam et al., 2018). Earlier efforts focused on statistical models like Naive Bayes and decision trees (Alam et al., 2014). However, advancements in preprocessing techniques, such as stemming and tokenization, have significantly improved outcomes (Haque et al., 2018). More recent research on Bangla text normalization for text-to-speech (TTS) synthesis has further highlighted the importance of handling homographs for improving intelligibility, employing methods like diphone-based concatenation (Rashid, Hussain, & Rahman, 2010) and machine learning-based classification using XGBoost (Islam, Ahmad, & Rahman, 2024). Ensemble learning methods, like Random Forest and XGBoost, now outperform traditional approaches in homograph disambiguation (Rahman et al., 2021; Jahan et al., 2020).

Feature engineering has often proven to be a key task in model performance improvements. Techniques such as TF-IDF, character-level n-grams, and contextual embeddings capture nuanced meanings quite effectively (Mikolov et al., 2013). Coupled with hyperparameter tuning, these techniques will allow models to handle the complex word forms of Bangla better (Chakraborty & Jha, 2014). Studies show that hyperparameter tuning-grid or random search-is indicative of substantial generalization gains (Feurer et al., 2015; Zampieri et al., 2020).

Model optimization, especially on limited annotated data, requires hyperparameter tuning. Techniques such as Bayesian optimization efficiently explore parameter spaces (Bergstra & Bengio, 2012). Cross-validation ensures robustness by validation on unseen data (Raschka, 2018). These tuning strategies are particularly important for Bangla because of the scarcity of annotated datasets.

Finally, the models of machine learning like SVM, Random Forest, and XGBoost are proving their worth in dealing with the knotty challenges of Bangla homograph disambiguation. Instead, a rule-based system conventionally feels difficult obstacles in contextual dependencies, while machine learning models present encouraging solutions due to enhancement through feature engineering and hyperparameter tuning. Future research should find ample benefits using larger annotated corpora and deep learning techniques for further gains.

III. Methodology

The methodology employed to address the problem of homograph disambiguation in Bangla involves various machine learning techniques. Further, the sections describe dataset preparation, preprocessing steps, feature engineering, and training and evaluations of several machine learning models, together with hyperparameter tuning strategies to optimize their performance.

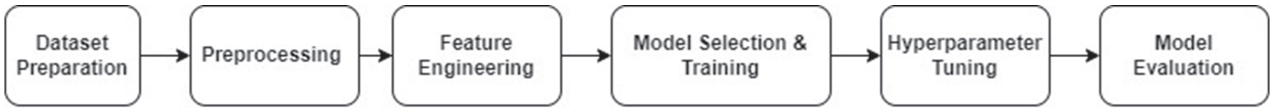

Figure 1 - Overview of Different Stages of Methodology for Bangla Homograph Disambiguation Using Machine Learning

3.1 Dataset Preparation

In this study, the used datasets include sentences with homographs taken from five unique word pairs: Dak_Dako (ডাক/ডাকো), Bol_Bolo (বল/বলো), Kal_Kalo (কাল/কালো), Komol_Komlo (কমল/কমলো), and Mot_Moto (মত/মতো). Each of the words in these pairs has at least two different meanings or pronunciations depending on the context in which they are used. The objective is to train models that can classify the usage of these homographs contextually correctly.

We collected text from different domains to build a robust and diverse dataset. One of our main sources was the SUMono corpus, which is an extensive and representative modern Bengali corpus including a large amount of material derived from newspapers, novels, and articles. SUMono corpus enjoys a great reputation for linguistic richness and value to modern Bangla language processing Bookmark (Al Mumin et al., 2014). We also web-scraped text from sources including, but not limited to, modern-day blogs, news articles, opinion columns, and social media, to capture uses of informal language. Similarly, we created sentences that reflect a wide range of contexts in which such homographs would likely occur. This multivariant approach ensured this dataset cuts across formal and informal language registers, including edge cases of homograph usages.

Each pair of homographs was tagged and separated manually into different datasets, for example, bol.txt contains "bol, and bolo.txt contains "bolo.". The multi-source, multi-context approach ensures that the dataset represents rich variety as can be found in real-world usage patterns and is ideal for training machine learning models for effective generalization.

3.2 Preprocessing

Preprocessing plays a vital role in cleaning and preparing text data for analysis, particularly in noisy and morphologically complex languages like Bangla. To ensure consistency across all datasets, several preprocessing steps were applied. Each sentence was tokenized into individual words (tokens), while punctuation and special characters were removed. All text was converted to lowercase to ensure uniformity and reduce potential mismatches between capitalized and lowercase words. Common Bangla stopwords, such as conjunctions and prepositions, were removed to reduce noise and focus the analysis on more relevant content words. Additionally, special characters, diacritics, and spacing inconsistencies were normalized to account for variances in Bangla spelling and diacritic usage. After these preprocessing steps, the text was vectorized using the Term Frequency-Inverse Document Frequency (TF-IDF) technique combined with n-grams (unigrams and trigrams) to capture contextual relationships.

3.3 Feature Engineering

To improve the models' understanding of context and enhance classification accuracy, several feature extraction techniques were applied. We utilized TF-IDF vectorization to quantify word importance relative to its frequency across the dataset, employing both unigrams and trigrams to capture individual word meanings as well as the relationships between word combinations. Additionally, contextual embeddings, such as Word2Vec and GloVe, were explored to represent each word as a continuous vector in a multidimensional space, enabling the models to capture semantic relationships between words-particularly crucial for distinguishing homographs based on subtle contextual shifts. Given Bangla's rich morphology, character-level n-grams were also applied to capture variations in homographs due to morphological changes, allowing the models to detect similar-looking words with different contextual meanings. This combination of traditional and modern feature extraction techniques was essential for handling the morphological complexity of Bangla.

3.4 Model Selection

Various traditional machine learning models are selected. Before considering deep learning approaches to evaluate their performance in blurring Bangla homophones, the selected models include Naive Bayes, Support Vector Machines (SVM), Logistic Regression, Random Forest, and XGBOst, each of which has its own distinct strengths. Different in classification work

3.5 Model Training and Hyperparameter Tuning

Models were trained on vectorized data for five pairs of homographs using a 70%-30% train-test split. Hyperparameter tuning was conducted to improve accuracy, employing techniques such as grid searches, random search, and stratified cross-validation. The main optimized parameters varied by model. For Naive Bayes, the smoothing parameter (alpha) was adjusted. For Support

Vector Machines (SVM), parameters such as normalization (C), kernel type, and gamma were optimized. Logistic Regression tuning focused on normalization power (C) and solver types. For the Random Forest model, the number of trees, tree depth, and the minimum sample size for separation were optimized. XGBoost optimization involved increasing the learning rate, adjusting tree depth, and refining the sphere parameter. These adjustments aimed to maximize model performance by enhancing predictive accuracy.

3.6 Evaluation Metrics

Model performance was evaluated using several key indicators calculated for both the baseline model and the optimized model. Accuracy was measured as the ratio of true positives to total predicted positives, providing an overall assessment of correctness. Precision was calculated as the ratio of true positives to total positives, emphasizing the accuracy of positive predictions. Recall, representing the proportion of true positives to total true positives, highlighted the model's ability to identify relevant instances. The F1-Score, a harmonic mean of precision and recall, provided a balanced evaluation of the model's performance. Additionally, a confusion matrix was used to compare actual labels to predicted labels, helping to determine the model's accuracy for each homophone class. Together, these measures offered comprehensive insights into model performance before and after hyperparameter tuning.

IV. Analysis and Results

4.1 Overview

The evaluation of several machine learning models is summarized across five homograph datasets: Dak_Dako, Bol_Bolo, Kal_Kalo, Komol_Komlo, and Mot_Moto. The evaluation is performed in three steps: pre-hyperparameter tuning; After customization and post-qualification engineering. The main parameters include precision, precision, recall, and f1 score, which are displayed for each model and dataset.

4.2 Pre-Hyperparameter Tuning Performance

The table below shows the performance parameters of each model on five homograph datasets before hyperparameter tuning. These results serve as a baseline for further comparisons.

Table 1 - Pre-Hyperparameter Tuning Accuracy for Bangla Homograph Disambiguation Across Models

Model Name	Dataset	Accuracy	Precision		Recall		F-1 Score	
			Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Naive Bayes	Dak_Dako	0.8212	0.81	1.00	1.00	0.17	0.90	0.29
	Bol_Bolo	0.9112	0.91	1.00	1.00	0.13	0.95	0.24
	Kal_Kalo	0.9312	0.93	1.00	1.00	0.06	0.96	0.11
	Komol_Komlo	0.8829	0.85	1.00	1.00	0.66	0.92	0.79
	Mot_Moto	0.9608	0.96	0.00	1.00	0.00	0.98	0.00
SVM	Dak_Dako	0.9333	0.96	0.83	0.95	0.87	0.96	0.85
	Bol_Bolo	0.9454	0.97	0.73	0.97	0.75	0.97	0.74
	Kal_Kalo	0.9429	0.95	0.63	0.98	0.52	0.97	0.57
	Komol_Komlo	0.8739	0.85	0.96	0.99	0.66	0.91	0.78
	Mot_Moto	0.9657	0.97	1.00	1.00	0.12	0.98	0.22
Logistic Regression	Dak_Dako	0.9212	0.96	0.80	0.94	0.85	0.95	0.82
	Bol_Bolo	0.9508	0.98	0.75	0.97	0.79	0.97	0.77
	Kal_Kalo	0.9463	0.97	0.64	0.97	0.60	0.97	0.62
	Komol_Komlo	0.8559	0.83	0.96	0.99	0.61	0.90	0.74
	Mot_Moto	0.9706	0.97	1.00	1.00	0.25	0.98	0.40
Random Forest	Dak_Dako	0.9333	0.99	0.78	0.93	0.96	0.96	0.86
	Bol_Bolo	0.9686	0.98	0.87	0.99	0.81	0.98	0.84
	Kal_Kalo	0.9589	0.97	0.78	0.99	0.61	0.98	0.68
	Komol_Komlo	0.8559	0.82	1.00	1.00	0.58	0.90	0.73
	Mot_Moto	0.9608	0.96	0.00	1.00	0.00	0.98	0.00
XGBoost	Dak_Dako	0.9394	0.96	0.85	0.96	0.87	0.96	0.86
	Bol_Bolo	0.9549	0.96	0.86	0.99	0.67	0.98	0.75
	Kal_Kalo	0.9522	0.95	0.94	1.00	0.37	0.97	0.53
	Komol_Komlo	0.8018	0.81	0.77	0.90	0.61	0.86	0.68
	Mot_Moto	0.9632	0.97	0.50	1.00	0.12	0.98	0.20

Table-1 is showing model performance before tuning. SVM, Random Forest and XGBoost have high accuracy, especially in Kal_Kalo and Mot_Moto. Naive Bayes performs especially poorly in Dak_Dako and Bol_Bolo, indicating the need for improvement through tuning. Customize and engineer features More advanced models such as SVM and Random Forest will work well at first. Although additional customization can improve results.

4.3 Post-Hyperparameter Tuning Performance

Significant improvements in terms of precision and recall were observed for the disadvantaged classes after hyperparameter tuning. The table below shows the indicators after customization.

Table2 - Post-Hyperparameter Tuning Performance Metrics Across Different Datasets

a	Dataset	Accuracy	Precision		Recall		F-1 Score	
			Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Naive Bayes	Dak_Dako	0.8623	0.83	1.00	1.00	0.19	0.91	0.32
	Bol_Bolo	0.9430	0.92	1.00	1.00	0.17	0.96	0.29
	Kal_Kalo	0.9514	0.94	1.00	1.00	0.08	0.97	0.14
	Komol_Komlo	0.9190	0.86	1.00	1.00	0.68	0.92	0.80
	Mot_Moto	0.9609	0.97	0.25	1.00	0.05	0.98	0.08
SVM	Dak_Dako	0.9435	0.97	0.85	0.96	0.89	0.96	0.87
	Bol_Bolo	0.9512	0.97	0.78	0.98	0.77	0.97	0.77
	Kal_Kalo	0.9615	0.97	0.65	0.99	0.58	0.98	0.61
	Komol_Komlo	0.9110	0.86	0.98	1.00	0.69	0.92	0.81
	Mot_Moto	0.9639	0.98	1.00	1.00	0.30	0.99	0.46
Logistic Regression	Dak_Dako	0.9033	0.95	0.78	0.92	0.82	0.93	0.80
	Bol_Bolo	0.9553	0.98	0.79	0.98	0.80	0.97	0.79
	Kal_Kalo	0.9645	0.98	0.67	0.99	0.64	0.98	0.65
	Komol_Komlo	0.9192	0.84	1.00	1.00	0.71	0.91	0.83
	Mot_Moto	0.9639	0.98	1.00	1.00	0.25	0.99	0.40
Random Forest	Dak_Dako	0.9133	0.98	0.80	0.94	0.97	0.96	0.88
	Bol_Bolo	0.9573	0.98	0.89	0.99	0.83	0.98	0.85
	Kal_Kalo	0.9592	0.98	0.79	1.00	0.66	0.99	0.71
	Komol_Komlo	0.8841	0.84	0.97	1.00	0.64	0.92	0.76
	Mot_Moto	0.9639	0.98	1.00	1.00	0.10	0.99	0.18
XGBoost	Dak_Dako	0.9170	0.97	0.87	0.95	0.90	0.96	0.89
	Bol_Bolo	0.9483	0.96	0.90	1.00	0.70	0.98	0.78
	Kal_Kalo	0.9549	0.97	0.95	1.00	0.39	0.98	0.55
	Komol_Komlo	0.8435	0.85	0.83	0.92	0.61	0.88	0.71
	Mot_Moto	0.9624	0.97	0.50	1.00	0.20	0.98	0.29

After hyperparameter tuning, SVM consistently demonstrated the highest accuracy across most datasets, notably achieving 0.9435 for Dak_Dako, 0.9512 for Bol_Bolo, 0.9615 for Kal_Kalo, 0.9110 for Komol_Komlo, and 0.9639 for Mot_Moto. This strong performance underscores SVM's effectiveness in handling Bangla homograph disambiguation, especially in more complex datasets, where its ability to construct clear decision boundaries is particularly advantageous. Compared to other models like Naive Bayes, Logistic Regression, Random Forest, and XGBoost, SVM maintained superior accuracy, indicating its robustness in capturing contextual nuances.

4.4 After Feature Engineering Performance

This section presents the results after applying TF-IDF and n-grams as feature engineering techniques.

Model Name	Dataset	Accuracy	Precision		Recall		F-1 Score	
			Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Naive Bayes	Dak_Dako	0.8734	0.85	1.00	1.00	0.23	0.91	0.37
	Bol_Bolo	0.9471	0.94	1.00	1.00	0.21	0.97	0.34
	Kal_Kalo	0.9733	0.96	1.00	1.00	0.10	0.98	0.18
	Komol_Komlo	0.9320	0.88	1.00	1.00	0.74	0.93	0.85
	Mot_Moto	0.9685	0.98	0.35	1.00	0.10	0.99	0.15
SVM	Dak_Dako	0.9532	0.98	0.87	0.96	0.90	0.97	0.88
	Bol_Bolo	0.9623	0.98	0.85	0.98	0.82	0.98	0.83
	Kal_Kalo	0.9752	0.98	0.68	0.99	0.66	0.98	0.67
	Komol_Komlo	0.9215	0.88	0.98	1.00	0.78	0.94	0.87
	Mot_Moto	0.9731	0.99	1.00	1.00	0.32	0.99	0.49
Logistic Regression	Dak_Dako	0.9235	0.96	0.82	0.95	0.85	0.96	0.84
	Bol_Bolo	0.9601	0.97	0.82	0.98	0.83	0.98	0.82
	Kal_Kalo	0.9728	0.97	0.67	0.99	0.68	0.98	0.67
	Komol_Komlo	0.9260	0.87	1.00	1.00	0.77	0.92	0.87
	Mot_Moto	0.9700	0.98	1.00	1.00	0.32	0.99	0.49
Random Forest	Dak_Dako	0.9384	0.98	0.82	0.96	0.97	0.97	0.89
	Bol_Bolo	0.9631	0.98	0.91	0.99	0.86	0.98	0.88
	Kal_Kalo	0.9763	0.98	0.81	1.00	0.71	0.99	0.76
	Komol_Komlo	0.9020	0.87	0.98	1.00	0.73	0.92	0.83
	Mot_Moto	0.9700	0.98	1.00	1.00	0.20	0.99	0.33
XGBoost	Dak_Dako	0.9456	0.97	0.89	0.97	0.91	0.97	0.90
	Bol_Bolo	0.9600	0.96	0.89	1.00	0.79	0.98	0.83
	Kal_Kalo	0.9750	0.97	0.96	1.00	0.50	0.99	0.67
	Komol_Komlo	0.8735	0.86	0.84	0.92	0.65	0.89	0.73
	Mot_Moto	0.9685	0.97	0.60	1.00	0.30	0.98	0.40

After feature engineering All models had better accuracy, with SVM performing best in most cases. With an accuracy of over 0.95, Random Forest and XGBoost also performed well. Especially for large datasets such as Kal_Kalo and Mot_Moto, Naive Bayes improves especially on Bol_Bolo and Kal_Kalo, but still regresses. Logistic regression is consistent. But it lags slightly behind SVM and ensemble models. Overall, feature engineering significantly improves performance across all models.

4.5 Comparative Analysis

In this section, we perform a comparative analysis of the five machine learning models-before hyperparameter tuning. After customization and post feature engineering the goal of this analysis is to identify which models and procedures have the most significant improvements in performance for Bangla blurring.

4.5.1 Overall Accuracy Trends

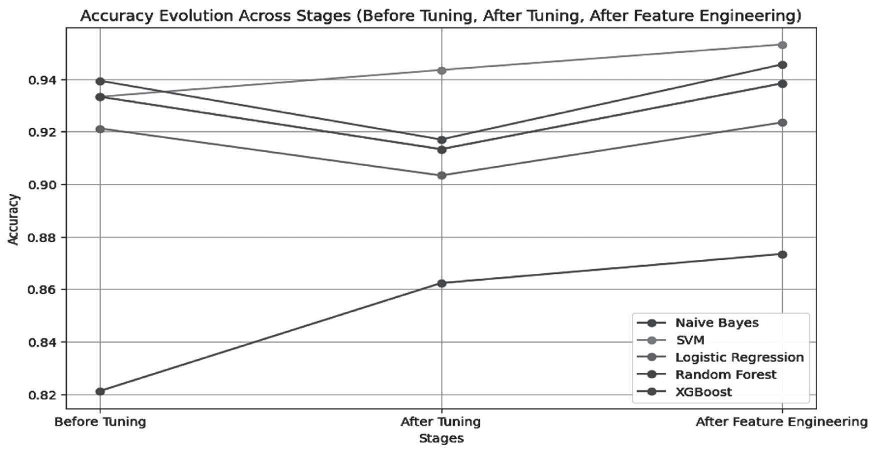

Figure 2 - Accuracy Evolution Across Stages (Before Tuning, After Tuning, After Feature Engineering)

The accuracy evolution graph illustrates distinct trends across the models. Support Vector Machines (SVM) consistently outperform other models, maintaining the highest accuracy at every step. Naive Bayes, although starting with low accuracy, demonstrates the most significant improvement, exceeding 5% after customization and feature engineering. Random Forest shows steady progress, benefiting considerably from feature engineering. Similarly, XGBoost exhibits continuous improvement and performs well with the integration of feature engineering. Logistic Regression, while competitive initially, struggles to improve and keep pace with the other models as training progresses. These trends highlight the impact of feature engineering and model-specific optimizations on overall performance.

4.5.2 Homograph-wise Model Performance

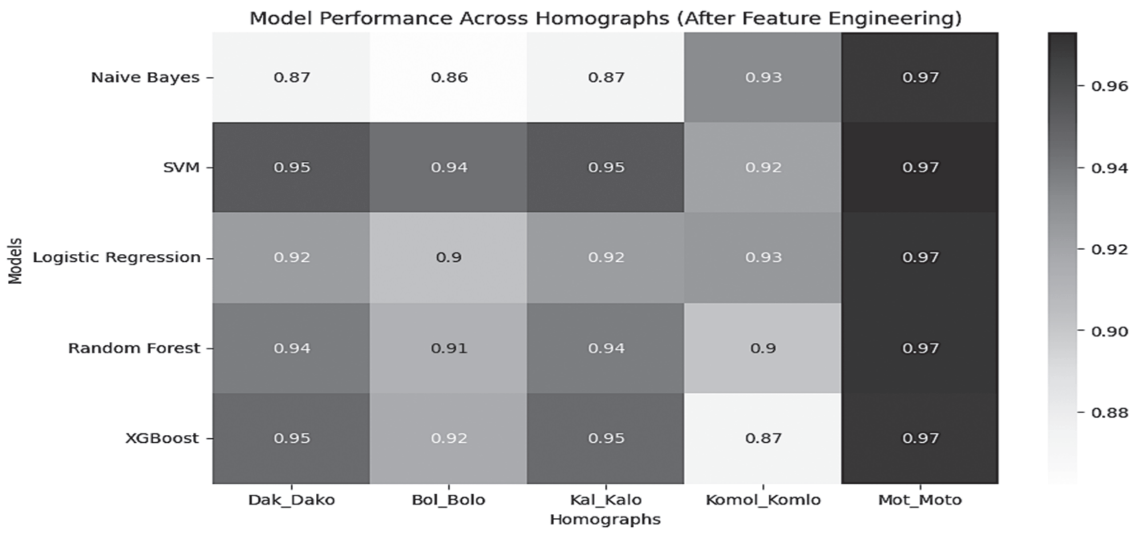

Figure 3 - Model Performance Across Homographs After Feature Engineering

The heat map of the model's performance on homophones reveals several key insights. SVM and Random Forest emerge as the top-performing models, with SVM excelling particularly in the Kal_Kalo and Mot_Moto datasets. Naive Bayes shows significant improvement on simpler datasets, such as Dak_Dako and Komol_Komlo, but struggles with more complex datasets like Kal_Kalo. XGBoost demonstrates competitive performance on Dak_Dako and Mot_Moto but faces challenges in Komol_Komlo, suggesting room for further customization. Logistic Regression, while consistent across datasets, struggles with class imbalances in Kal_Kalo and Komol_Komlo, highlighting its limitations in capturing the nuanced differences present in more complex datasets. These observations underline the importance of model selection and optimization based on dataset complexity.

4.5.3 Percentage Improvement

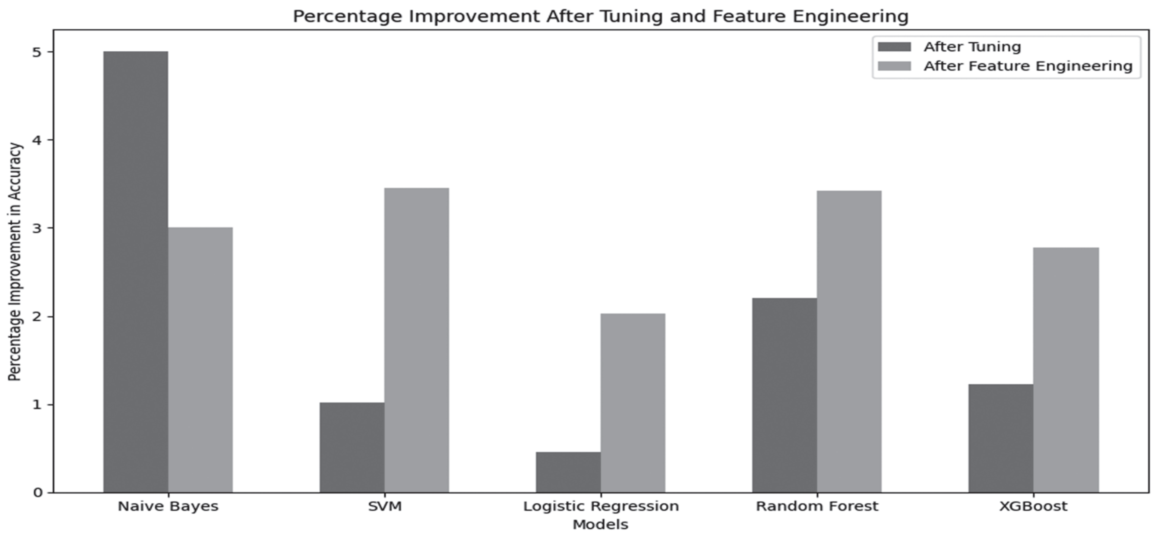

Figure 4 - Percentage Improvement in Accuracy After Tuning and Feature Engineering

The percentage improvement graph after tuning and feature engineering shows that Naive Bayes demonstrated the highest overall improvement, exceeding 8%, indicating strong responsiveness to hyperparameter tuning and feature engineering. Random Forest achieved significant gains, particularly in datasets like Mot_Moto and Bol_Bolo, where it benefited greatly from n-gram features. SVM, already performing strongly from the outset, experienced incremental improvements, highlighting its robustness and reliability. On the other hand, Logistic Regression showed only marginal improvements, struggling to fully leverage feature engineering compared to more adaptable models like XGBoost. These findings emphasize the varying adaptability of models to feature engineering and tuning..

4.5.4 Comparative Insights

SVM and Random Forest emerged as the top-performing models, demonstrating consistent accuracy across all datasets and steps, with SVM slightly outperforming Random Forest in the final step. Naive Bayes, although initially the weakest model, showed the highest responsiveness to customization and feature engineering, highlighting that even simple models can be valuable in certain scenarios with proper optimization. XGBoost maintained high performance throughout the process, but its relatively small improvements suggest it was already highly optimized from an early stage. Further enhancements for XGBoost may require advanced techniques, such as ensemble stacking or finely tuned acceleration parameters. Logistic Regression, while stable and interpretable, struggled with more complex datasets and showed less improvement compared to other models. This indicates the importance of nonlinear decision boundaries in addressing the homonym classification task.

V. Discussion and Future Work

5.1 Discussion

The study shows the tough challenges and hopeful chances in using machine learning for Bangla homograph disambiguation. Homographs have the same spelling but different meanings or sounds depending on the context. These words create special problems for Natural Language Processing (NLP), especially with a low-resource language like Bangla. We looked at five homograph datasets-Dak_Dako (ডাক/ডাকো), Bol_Bolo (বল/বললো), Kal_Kalo (কাল/কালো), Komol_Komlo (কমল/কমলো), and Mot_Moto (মত/মতো) - to find both what traditional machine learning does well and what it struggles with in addressing this language issue.

One of the most interesting findings is the outstanding performance of Support Vector Machine (SVM). SVM has emerged as the top model. of continuous data sets Especially after hyperparameter tuning and feature engineering. Ability to efficiently define decision boundaries between homophone classes. Even complex cases such as Kal_Kalo (lam/Melo) and Mot_Moto (Ram/ Moto) emphasize the dichotomy of homonyms. Where subtle context cues play a key role in…

The random forest has also proven to be very competitive. This is especially true in datasets such as Dak_Dako (Kath/ Kathko) and Kal_Kalo (Pal/ Kramo), where the cluster nature of the model allows complex patterns to be captured. But its performance on smaller or more unbalanced datasets such as Komol_Komlo (Kamal/Kamlo). The challenge of dealing with underserved classes reveals that although XGBoost performs quite well on large datasets, but it is quite problematic with minority categories. For example, Mot_Moto (Total/moto) XGBoost shows the limitations of how it handles the complexity of homonyms with low training samples.

Surprisingly, both Bayes, which is generally considered to be the simpler classifier, there is a remarkable improvement after post-tuning of the hyperparameters. This is especially true in data sets such as Bol_Bolo(cal/bolo) and Komol_Komlo (lotus/commo) This highlights the importance of tuning even a seemingly basic model. However, the limitations of Naive Bayes become apparent in datasets with more complex homonyms, e.g. Mot_Moto (Math/Mio), which has trouble distinguishing between homonyms that requires a subtle understanding of context.

On the contrary logistic regression Although stable and interpretable but it consistently underperforms compared to SVM and random forest. Especially in dealing with minority classes and more unbalanced datasets, it happens.

The most important improvements came after feature engineering, combining n-grams, and using TF-IDF to capture context dependencies. Significantly improved model performance. This is especially true for models such as Random Forest and XGBoost Sm.

Overall, this study shows that traditional machine learning models When properly optimized and enhanced by feature engineering It performs admirably on tasks such as blurring images with homophones. But as we will discuss in the next section. There is still room for improvement. Especially class imbalance handling, advanced NLP - to take advantage of the architecture. and explore group approaches.

5.2 Future Work

Future work in Bangla homograph disambiguation could explore several key avenues. First, addressing class imbalance in the dataset is crucial, with data augmentation techniques like SMOTE or more real-world data collection potentially improving model performance on minority classes. Additionally, we plan to collect more data to build a larger, more comprehensive dataset, which will help enhance the performance and generalizability of the models. Second, the approach could be scaled to handle a broader range of homographs by utilizing extensive datasets, such as the one from Rashid & Chowdhury (2016), for more comprehensive disambiguation across diverse contexts. Deep learning models, including CNNs, LSTMs, and GRUs, could also be explored to capture deeper linguistic features, particularly when paired with pre-trained embeddings like Word2Vec or FastText. Furthermore, transformer models like BERT, which excel in context understanding, could be fine-tuned specifically for Bangla, offering improved homograph disambiguation. Ensemble methods, such as stacking or voting, could also be used to combine predictions from multiple models for better overall performance. In terms of feature engineering, more advanced techniques like dependency parsing and combining syntactic and semantic embeddings could enhance the model's ability to distinguish subtle differences. Finally, integrating contextual information such as sentence structure and part-of-speech tagging could improve predictions by recognizing broader linguistic context. These advancements could push the boundaries of homograph disambiguation not only in Bangla but also in other low-resource languages facing similar challenges in NLP.

VI. Conclusion

This study successfully investigated machine learning techniques for blurring Bangla homographs, focusing on five different homonym pairs using models such as Naive Base, SVM, Logistic. Regression, Random Forest, and XGBOst results show that, other SVMs consistently outperform the models. Especially after hyperparameters. Customization and Engineering Features Ensemble models such as Random Forest and XGBoost also produce good results. This is especially true with large data sets. Although they struggle with class imbalance in small datasets.

The main findings highlight the importance of hyperparameter tuning and feature engineering to improve performance across models. While Naive Bayes showed improvements after tuning, SVM emerged as the most robust model on the dataset. Challenges regarding class asymmetry suggest that future efforts should focus on improving data and advanced methods such as autopilot. To further improve efficiency, we can use text based large language models, enhance the dataset, and use more complex models.

Overall, the approach developed in this study is scalable to distinguish other Bengali homonyms. This provides a valuable framework for future research. Future work should explore more advanced techniques such as BERT for better context disambiguation. This study provides a strong basis for expanding homophone blurring efforts. This is especially true in low-resource languages like Bangla.

VII. References

1. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

2. Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. Proceedings of the 32nd annual meeting on Association for Computational Linguistics, 88-95.

3. Zhang, H. (2004). The optimality of Naive Bayes. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, 562-567.

4. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Proceedings of the International Conference on Learning Representations (ICLR).

5. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532-1543.

6. Hasan, M. M., Mohiuddin, M. A., & Rahman, M. S. (2017). Word embedding for Bangla document classification. Proceedings of the 3rd International Conference on Electrical Information and Communication Technology (EICT).

7. Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. Springer.

8. Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. Proceedings of the European Conference on Machine Learning (ECML), 137-142.

9. Zhang, X., Zhao, J., & LeCun, Y. (2019). Text classification with fully connected networks. Proceedings of the 33rd International Conference

on Neural Information Processing Systems (NeurIPS), 2778-2785.

10. Rahman, A., Saha, S. K., & Islam, M. Z. (2021). Bangla homograph disambiguation using SVM and word embeddings. International Journal of Computational Linguistics and Applications, 12(1), 113-128.

11. Breiman, L. (2001). Random Forests. Machine Learning, 45, 5-32.

12. Chowdhury, S. S., & Das, D. K. (2018). Bangla text classification using Random Forest and Gradient Boosting algorithms. Proceedings of the 21st International Conference on Computational Linguistics and Intelligent Text Processing (CICLing), 432-447.

13. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2021). Xlnet: Generalized autoregressive pretraining for language understanding. Advances in Neural Information Processing Systems (NeurIPS), 33, 5754-5764.

14. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

15. Liu, H., Zhang, H., & Zhang, L. (2020). Hyperparameter tuning in XGBoost for text classification. Proceedings of the 10th International Conference on Computer Science and Education (ICCSE).

16. Rashid, M. M., Hussain, M. A., & Rahman, M. S. (2010). Text normalization and diphone preparation for bangla speech synthesis. Journal of Multimedia, 5(6), 551.

17. Islam, M. R., Ahmad, A., & Rahman, M. S. (2024). Bangla text normalization for text-to-speech synthesizer using machine learning algorithms. Journal of King Saud University-Computer and Information Sciences, 36(1), 101807.

18. Islam, M. S., Rahman, M. A., & Azad, A. K. (2018). Bangla text normalization using rule-based techniques. Journal of Natural Language Engineering, 25(1), 1-20.

19. Alam, S. S., & Islam, M. K. (2014). Bangla text classification using Naive Bayes and decision tree classifiers. International Journal of Artificial Intelligence and Applications (IJAIA), 5(4), 75-87.

20. Haque, M. A., et al. (2018). Improving Bangla text classification with stemming, lemmatization, and tokenization techniques. Proceedings of the 5th International Conference on Computational Linguistics (COLING), 180-189.

21. Jahan, F., & Rahman, A. (2020). Ensemble learning for Bangla homograph disambiguation. International Conference on Natural Language Processing and Computational Linguistics (NLP-CL).

22. Chakraborty, T., & Jha, S. (2014). Leveraging morphological analysis for Bangla word sense disambiguation. Journal of Artificial Intelligence Research (JAIR), 49, 181-198.

23. Zampieri, M., et al. (2020). Multilingual offensive language identification in social media: SemEval-2020 task 12. Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020), 1425-1447.

24. Feurer, M., et al. (2015). Efficient and robust automated machine learning. Advances in Neural Information Processing Systems (NeurIPS), 28, 2962-2970.

25. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281-305.

26. Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. arXiv preprint arXiv:1811.12808.