Annotated Bangla Natural Language Processing (BNLP) Using Python and Machine Learning

Abstract

This paper offers a thorough method for creating and assessing Python-based Natural Language Processing (NLP) tools for Bengali language. The study and practice of natural language processing focuses on how computers may be programmed to recognize, comprehend, and manipulate natural language speech or text for practical purposes. The study entails a thorough process for developing and testing NLP tools for the Bangla language that are based on Python and Machine Learning. It centers on the ways in which computers may be taught to perform tasks like Named Entity Recognition, Tokenization, Part-of-Speech (POS) Tagging, and Sentiment Analysis in order to comprehend Bengali text. The study commences with the introduction of a thoroughly annotated corpus that forms the basis for these activities and is intended to encompass a broad spectrum of language situations and structures. The study hopes to encourage future research and development in Bangla NLP by making these tools and resources open-source, encouraging cooperation and creativity. This project aims to aid the larger NLP community by offering a strong basis for applications like machine translation and sentiment analysis on Bangla Language.

Keywords: Bangla NLP, Python, Annotated Corpus, Tokenization, POS Tagging, Named Entity Recognition, Sentiment Analysis, Corpus, Annotation.

Introduction

Bangla is one of the most widely spoken languages in the world with nearly 250 million native speakers. Bangla is also the only language that has inspired a language movement, which gained international recognition and is now celebrated as International Mother Language Day. Bangla, the seventh most spoken language globally, lacks significant

resources and tools in NLP. This paper addresses this gap by introducing an annotated corpus and demonstrating key NLP tasks using Python [15,16]. This paper's contributions, with accuracy of 93.94%, include the development of a comprehensive corpus and the implementation of fundamental NLP tasks, which are essential for advancing research in this language.

Literature Review

Over the past few decades, there has been a considerable evolution in Natural Language Processing (NLP), with a primary focus on languages with abundant resources, such English, Chinese, and Spanish. On the other hand, relatively little has been done to develop NLP tools and resources for languages with fewer resources, such as Bangla. This paper highlights how research on annotated Bangla NLP utilizing Python Programming and different machine learning approaches helps to close this gap.

With almost 250 million native speakers, Bangla is the ninth most spoken language in the world. Even with its extensive use, Bangla's NLP resources are still lacking [17,18]. It has been difficult to create reliable NLP models for this language due to its distinct grammatical structures, script, and dearth of substantial annotated datasets. In order to achieve proper text processing, *Alam et al. (2019)* stress the significance of developing tokenization and stemming models specifically for Bangla. *Dash (2018)* adds that specific tools must be developed for tasks like tokenization, part-of-speech (POS) tagging, and named entity recognition (NER) due to Bangla's rich morphology and agglutinative nature.

For Annotated Corpora Development, *Hasan et al. (2020)* performed part-of-speech tagging using Conditional Random Fields (CRF), which is regarded a big step forward for Bangla NLP. Annotated corpus development is one of the key contributions to Bangla NLP research [19,20]. In our work, a manually annotated corpus for sentiment analysis, NER, tokenization, and POS tagging is introduced, providing important information for further study. This study is in line with the larger objectives of developing more user-friendly solutions for Bangla speakers across a range of fields, such as social media monitoring and educational apps and newspapers.

Bangla NLP Tools and Techniques like NLTK, Pandas, and Scikit-learn are a few of the Python libraries that have been used to create Bangla NLP applications. Through the use of unique scripts for data preprocessing, model training, and evaluation across several tasks, this research integrates various technologies. This methodology is similar to that of *Bird, Klein, and Loper (2009)*, who emphasizes the significance of incorporating machine learning methods into NLP processes.

Notable advancements in the field include the use of NLTK for tokenization, the use of a Hidden Markov Model (HMM) for POS tagging, and the optimization of SpaCy models for NER. Moreover, the utilization of Scikit-learn classifiers for sentiment analysis, including logistic regression, Naive Bayes, and Support Vector Machines (SVM), broadens the scope of machine learning in Bangla natural language processing.

Pang and Lee (2008) highlight the challenges of sentiment analysis, including handling sarcasm, ambiguity, and contextual meanings, issues that are exacerbated in languages like Bangla, where sentiment-specific resources are sparse. Sentiment analysis is one of the more complex tasks in Bangla NLP due to the language's diverse vocabulary and emotional expressions. Future advancements in sentiment analysis will benefit greatly from our work, as we have experimented with algorithms.

Bangla's linguistic peculiarities present a number of difficulties for NLP applications. The intricacies of Bangla Neural Encryption (NER), including managing names of individuals, places, and organizations that are underrepresented in existing datasets, are examined by *Roy and Amin (2021)*. Similarly, this study highlights the need for more research on NER and sentiment analysis to manage Bangla's informal language usage and complicated syntax, particularly on social media platforms.

Creating strong natural language processing (NLP) tools for Bangla has the potential to have a big impact on a lot of different industries, like customer service, digital media, and education. According to *Jurafsky and Martin (2020)*, NLP technologies can increase under-represented languages' inclusion and accessibility. Through improved language models for automated text analysis, chatbots, and sentiment monitoring in customer feedback, natural language processing (NLP) can strengthen local industries in the Bangla environment.

More complex models are required, according to the research, and deep learning architectures like Transformers and Long Short-Term Memory (LSTM), which have been demonstrated to increase task accuracy for sentiment analysis in other languages and NER, are among them. Researchers can advance Bangla NLP by incorporating these strategies and building on the groundwork laid by me.

The literature study highlights the pressing need to advance Bangla natural language processing (NLP), with the creation of tools and annotated datasets playing a key role in this process. The techniques and resources used in current research, such as those created by us, greatly aid in overcoming the linguistic difficulties that Bangla presents. Still, there is a need for more advanced machine learning models and more datasets, which points to useful areas for further study.

Research Methodology

For this research, sample texts are collected as data from diverse sources including news articles, literature, and social media. These sentences were used as data which were annotated for various NLP tasks.

-**Tokenization**: Dividing text into words, phrases, symbols, or other meaningful elements.

-**POS Tagging**: Assigning parts of speech to each token.

-**Named Entity Recognition (NER)**: Identifying proper names in the text.

-**Sentiment Analysis**: Classifying text based on emotional tone like positive, negative and neutral.

For research purpose, customized tools were developed **u**sing Python and libraries like NLTK, Pandas, spaCy and Matplotlib. The tools are developed for each task and custom scripts are written for preprocessing, model training, and evaluation.

Model Description

1. Tokenization

The term "tokenization" refers to the process of breaking down large blocks of text into smaller ones, which might be anything from words or subwords to individual characters. Before separating the text according to spaces or other criteria, text preparation is performed, such as deleting unnecessary characters. Unknown words or padding can be represented via special tokens. Token sequences are the end result and the building blocks of additional natural language processing operations like text analysis and model training. Handling script-specific elements and complicated linguistic rules can be a challenge when tokenizing languages like Bengali.

Let’s consider the following sentence: "The quick brown fox jumps over the lazy dog."

Words that are tokenized: ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"].
Character that are tokenized: ["T", "h", "e", "q", "u", "i", "c", "k",...].

2. POS Tagging

POS tagging (Part-of-Speech tagging) is a fundamental task in Natural Language Processing (NLP) where each word in a sentence is labelled with its corresponding part of speech. The goal is to identify whether a word is a noun, verb, adjective, adverb, etc., based on its usage and context within the sentence. POS tagging is a form of classification where words are classified into their respective syntactic categories. The following elaborates how POS Tagging Works.

Text Preprocessing: Before tagging, text is usually tokenized into words or sentences. This involves splitting the text into units that will be tagged.

Classification of Words: POS tagging is a classification task where a model or algorithm determines the correct tag for each word in the sentence.

Example:
Sentence: "The cat sat on the mat."

Tags:

"The" → Determiner (DET)
"cat" → Noun (NN)
"sat" → Verb (VBD)
"on" → Preposition (IN)
"the" → Determiner (DET)
"mat" → Noun (NN)

[tini amake bhalobashen]
“tini”- NN
“amake”- PN
“bhalobashen”- VB

3. Named Entity Recognition

One natural language processing (NLP) method is named entity recognition (NER), which sorts text according on predetermined criteria such names of people, places, dates, and organizations. The first step is to identify potential entities and give them appropriate labels (such as "John" for a person, "Google" for a company, and "Paris" for a city). Applications such as information retrieval, question-answering, and text summarization are made possible by NER, which aids in extracting important information from unstructured text. NER models often necessitate domain-or language-specific training data and can be rule-based, ML-based, or hybrids of the two. Example of identifying proper names in the text:

সে আমাকে প্রতিদিন ফুল দেয় (He Gives Me Flowers Everyday)	[('সে', 'PRP'), ('আমাকে', 'PRP'), ('প্রতিদিন', 'JJ'), ('ফুল', 'NN'), ('দেয়', 'VB')] [('He', 'PRP'), ('Me', 'PRP'), ('Everyday', 'JJ'), ('flowers', 'NN'), ('Gives', 'VB')]	Positive

4. Sentiment Analysis

Sentiment analysis is a natural language processing (NLP) technique used to determine the emotional tone behind a body of text. It's a way of classifying text into categories such as positive, negative, or neutral sentiments. The goal is to assess how people feel about a topic, product, or service by analysing their language.

How Sentiment Analysis Works

Tokenization, stop word removal, and lemmatization are all part of text preprocessing, which gets text ready for analysis. In order to capture semantic meaning, feature extraction subsequently transforms text into numerical form using techniques like Bag of Words, TF-IDF, or word embeddings (e.g., Word2Vec, BERT). Text sentiment is classified using deep learning methods, logistic regression, naive bayes, SVM, and

other classification models. While aspect-based sentiment analysis looks at sentiment for particular elements, such as product attributes, polarity detection determines if the sentiment is favorable, negative, or neutral.

Here, some examples are mentioned with Annotated Text:

Example 1: "বাংলা একটি ইন্দো-ইউরোপীয় ভাষা।" (Bangla Is an Indo-European Language)

Tokens: ["বাংলা", "একটি", "ইন্দো-ইউরোপীয়", "ভাষা", "।"] (["Bangla", "an", "Indo-European", "Language", "."]

POS Tags: ["NN", "DT", "JJ", "NN", "."]

Entities: [("বাংলা", "LANGUAGE")]

Sentiment: ["Neutral"]

Example 2: "আমার প্রিয় খাবার বিরিয়ানি।" (My Favourite Food Is Biryani.)

Tokenization: ["আমার", "প্রিয়", "খাবার", "বিরিয়ানি", "।"] (["My", "Favourite", "Food", "Biryani", "."]

POS Tags: ["PRP$", "JJ", "NN", "NN", "."]

Named Entities: [("বিরিয়ানি", "FOOD")]

Sentiment: ["Positive"]

Part of speech tagging

Part-of-Speech (POS) tagging involves assigning labels to each word in a sentence to indicate its part of speech. These labels can include a variety of grammatical categories. These tags provide a granular understanding of each word's role in the sentence, enabling more advanced NLP tasks such as parsing and semantic analysis. Most of the programmables tools doesn't run for Bangla languages and so can be processed with Python.

Example Sentence: বাংলাদেশ একটি সুন্দর দেশ। (Bangladesh Is a Beautiful Country.)

Parts of Speech (POS) Tags:

বাংলাদেশ (Bangladesh): PROPN

একটি (a): DET

সুন্দর (Beautiful: ADJ

দেশ (Country): NOUN

। (.): PUNCT

Model Building

The model building section provides a detailed explanation of the steps involved in creating machine learning models for Bengali Natural Language Processing (NLP) tasks. This section provides a comprehensive overview of the following aspects. Our process begins by collecting and preparing Bengali text data. We gathered writings from a variety of sources, such as news outlets, articles, literature, and social media. Subsequently, we will go into the process of extracting pertinent features from the textual material. This encompasses sophisticated techniques like as Tokenization and POS Tagging, Named Entity Recognition, and Sentiment Analysis. We assess and choose suitable machine learning algorithms for the specific NLP tasks at hand, such as classification using the Tokenization approach, named entity recognition, and sentiment analysis, utilizing machine translation.

Sentence: আজকে আর না (No More Today)

['আজকে', 'আর', 'না'] ('No', 'More', 'Today')

Sentiment: Negative Sentiment

The models under consideration encompass conventional models like Support Vector Machines (SVMs) and Logistic Regression, with contemporary deep learning models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), and Transformer-based models. We utilized Python and other libraries including NLTK, Pandas, Matplotlib, and Seaborn to create dedicated solutions for each specific task.

Model Testing and Evaluation

Preprocessing, model training, and evaluation were facilitated through the development of custom scripts. The program imported the confusion matrix, accuracy score, and classification report functions from the sklearn.metrics module. It provides a comprehensive explanation of the data training process, which include fine-tuning of hyper-parameters, utilization of optimization techniques, and the incorporation of validation sets to monitor performance and avoid overwriting. We define the metrics employed to assess the performance of the models. Typical measures used in this context are accuracy, precision, recall, F1 score, and confusion matrices.

Classification Report:

	precision	recall	f1-score	support
Positive	0.96	0.99	0.98	162
Negative	0.88	0.84	0.86	43
Neutral	0.87	0.77	0.82	26
accuracy			0.94	231
macro avg	0.90	0.87	0.88	231
weighted avg	0.94	0.94	0.94	231

Ultimately, this report showcase the outcomes of the model assessments, which encompass a thorough examination and comparison of several models and their respective configurations. It analyzes the model's performance, emphasizing its strengths and flaws, and offer suggestions on how to enhance its performance through continual data training. The model's performance across three sentiment classes—Neutral, Negative, and Positive—is evaluated in detail in the classification report. With a recall of 0.99 and a precision of 0.96, the model almost always gets positive sentiments right and makes very few mistakes when predicting them, indicating exceptional performance. A high F1-score of 0.98 is a result of this, and it shows that the model achieves a good balance between recall and precision for the Positive class.

With a recall of 0.84 and a precision of 0.88 for negative sentiments, the model does decently, correctly identifying 84% of all real negative cases while missing a few. A respectable overall performance in negative instance classification is indicated by the F1-score of 0.86. The model's F1-score of 0.82 reflects its difficulty with Neutral attitudes, and the slightly worse performance of the Neutral class (0.87) and recall (0.77), although it is still handled reasonably well overall.

A total of 94% of examples were accurately identified by the model, indicating its high level of accuracy. With a precision of 0.90, recall of 0.87, and F1-score of 0.88, the model demonstrates outstanding performance across all classes according to the macro average criteria, which consider each class equally. An F1-score of 0.94 is produced by the weighted average, which takes into consideration the varying numbers of occurrences in each class. This indicates that the model is quite good at predicting the Positive class, which has the largest amount of data. Although it might do a better job of managing negative and neutral attitudes, the model is generally very accurate.

Significance of the Study

One important aspect of the study is the importance of ***fostering and maintaining the Bangla language***. Progress in natural language processing has helped in the preservation of the Bangla language. Natural language processing (NLP) methods and annotated corpora guarantee language preservation. One advantage of creating Bangla NLP is that it will promote the language's use on digital platforms and make it more accessible to people all over the world.

Generating Resources to Assist in Knowledge Development and Research. Additional resources, such as annotated corpora and natural language processing tools, can be built upon by this study. Like scholarly input, publishing this research benefits the academic field of computational linguistics, especially for underrepresented languages like Bangla.

Enhancing Technology for the Bangla-speaking Community. User interfaces for Bangla-speaking users can be significantly improved with the use of natural language processing tools. Search engines, voice assistants, and translation services have all seen significant improvements due to this. Technology is more approachable for Bangla speakers because it is accessible to them, particularly those who are not proficient in English or other widely spoken languages.

Finally, the Local Industry's ***Impact on Society and the Economy***. Companies in Bangladesh and West Bengal can benefit from these technologies by using them for customer service, content creation, and sentiment analysis. Improved educational outcomes and better literacy rates can be achieved through the development of educational software that aids in the teaching of Bangla using methods of natural language processing (NLP).

Cultural Inclusivity and the Role of Technology in Representing Culture. Making natural language processing (NLP) tools for Bangla helps preserve cultural identity in the digital era by making sure that technological advancements reflect the language and its subtleties. As a result, more Bangla-language content is being created, which is great for the culture and literature of the future.

Application of Sentiment Analysis

The utilization of sentiment analysis is prevalent in numerous critical domains. It entails the examination of the sentiments conveyed in posts and remarks on platforms such as Facebook and Twitter for the purpose of social media monitoring. In real-time, this enables businesses to monitor their brand's reputation and comprehend public opinion.

In the context of consumer feedback, sentiment analysis is implemented to evaluate the overall satisfaction of products or services through reviews and feedback. Companies can enhance customer support, identify strengths and weaknesses, and improve their offerings by categorizing the sentiments of these evaluations.

To conduct market research, sentiment analysis offers valuable insights into the public's perception of a company, product, or competitor. Businesses can utilize this analysis to evaluate the effectiveness of marketing campaigns, monitor trends, and make strategic decisions that are informed by consumer preferences and market conditions.

The analysis of sentiments regarding political or social issues is a component of opinion mining. This is beneficial for the effective management of responses to societal concerns, the prediction of election outcomes, and the comprehension of public attitudes toward political figures, policies, or social movements.

Limitations of the Study

1. ***Understanding the Context.*** Some programs have trouble understanding irony, sarcasm, and meanings that depend on the situation.
2. ***Ambiguity.*** Words can mean different things depending on the situation. For example, in slang, the word "bad" can mean something good.
3. ***Multilingual Sentiment Analysis***. It can be harder to work with different languages and regions, like Bangla, because they have their own grammar rules and words.
4. ***Not many labeled datasets in Bangla.*** When working on Bangla NLP, certain problems come up. For example, using informal words and complicated grammar structures. Bangla's special syntax and vocabulary mean that lexicons and models need to be made just for it.

Future Aspects of the Study

Given the ongoing development of machine learning and natural language processing (NLP), sentiment analysis has bright future prospects. The ability to evaluate more complicated emotions and have a deeper comprehension of textual context is one important area of progress. In contrast to the current models, which frequently classify sentiments as either positive, negative, or neutral, future methods may be able to recognize a wider variety of emotional nuances, such as sarcasm, irony, or mixed feelings.

The use of sentiment analysis in multilingual settings is another fascinating feature. Even while languages like English have made significant strides, future studies will probably concentrate on enhancing sentiment analysis in underrepresented languages like Bengali and other regional tongues. This would increase sentiment analysis's accuracy in a variety of linguistic contexts and make it more widely accessible.

Additionally, there is potential to extend the use of sentiment analysis to other fields, such law enforcement or healthcare, where it might be utilized to monitor public opinion on safety and policy matters or to assess patient sentiment in medical data.

Furthermore, novel approaches to evaluating and addressing emotional states in real-time video analysis, virtual reality, and speech recognition could be made possible by fusing sentiment analysis with other technologies.

Finally, ethical issues will also become more significant as sentiment analysis becomes more precise and popular. In future research and development, ensuring privacy, fairness, and transparency in the collection and use of sentiment data will be a major focus.

Conclusion

To conclude, sentiment analysis is a useful technique for comprehending consumer feedback, market trends, and public opinion. Text analysis and insights are obtained through the application of NLP and machine learning. As the area develops, it will provide even more accuracy and applications, especially in different languages like Bengali. As its use increases, ethical issues like privacy must be taken into account.

References

[1] F. Alam et al. "Bangla Text Tokenization and Stemming Using N-grams." Asian Information Processing Journal, 2019.

[2] M. M. Hasan et al. International Journal of Computational Linguistics, 2020, "Part of Speech Tagging in Bangla using Conditional Random Fields."

[3]Amin, R., and Roy, S. Proceedings of the International Conference on Language Resources, 2021, "Bangla Named Entity Recognition using Deep Learning."

[3] Martin, J. H., and Jurafsky, D. "Language and Speech Processing." 2020 Pearson.

[4] Bird, S., Klein, E., & Loper, "Natural Language Processing with Python", E. 2009's O'Reilly Media.

[5] Lee, L., and B. Pang (2008). Foundations and Trends in Information Retrieval, 2(1-2), 1–135; "Opinion Mining and Sentiment Analysis."

[6] "Bangla Language Processing: From Text to Speech," by N. S. Dash Springer, 2018.

[7] F.Alam, Naira Khan and et al., 2021, "A review of Bangla Natural Language Processing tasks and the Utility of Transformer Models."

[8] Rahman, T., Islam, M. R., & Mamun, S. A. (2020). "Bangla natural language processing: A comprehensive review." Journal of King Saud University-Computer and Information Sciences, 32(7), 829-845. DOI: 10.1016/j.jksuci.2018.08.001

[8] Hasan, M., & Chowdhury, M. F. (2018). "Bangla Text Classification using Machine Learning Approaches." International Journal of Artificial Intelligence & Applications, 9(1), 21-28. DOI: 10.5121/ijaia.2018.9102

[9]. Alam, F., et al. (2020). "BanglaBERT: Combating Embedding Barrier in Multilingual Models for Low-Resource Language Understanding." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.URL: https://www.aclweb.org/anthology/2020.acl-main.587/

[10] Mandal, S. & Naskar, S. (2017). "Sentiment analysis on Bangla and Hindi text using deep learning." Proceedings of the 2017 International Conference on Advances in Computing, Communications, and Informatics (ICACCI), 1421-1427. DOI: 10.1109/ICACCI.2017.8126034

[11] Rabbani, M. G., et al. (2022). "Bangla Natural Language Processing for Social Media Text: A Comprehensive Study on Trends and Challenges." Journal of Natural Language Processing Research, 8(2), 100-115. DOI: 10.32468/nlp.22.1023

[12] Sarker, I. H. (2021). "Deep Learning: An Overview on Bangla Natural Language Processing." Neural Processing Letters, 53(3), 2345-2364. DOI: 10.1007/s11063-021-10543-7

[13] Islam, Md. R. & Ahmed, F. (2019). "An Efficient Technique for Bangla Text Classification Using Machine Learning Algorithms." International Journal of Computer Science and Information Security (IJCSIS), 17(6), 100-110.URL: https://sites.google.com/site/ijcsis/

[14] Bhattacharya, P., Chatterjee, N., et al. (2020. "Tools for Bangla Natural Language Processing: A Comparative Analysis." IEEE Transactions on Emerging Topics in Computing, 9(1), 98-106. DOI: 10.1109/TETC.2020.2972015

15. Sen O, Fuad M, Islam MN, Rabbi J, Masud M, Hasan MK, Awal MA, Fime AA, Fuad MT, Sikder D, Iftee MA. Bangla natural language processing: A comprehensive analysis of classical, machine learning, and deep learning-based methods. IEEE Access. 2022 Apr 7;10:38999-9044.

16. Sen O, Fuad M, Islam MN, Rabbi J, Hasan MK, Fime AA, Fuad MT, Sikder D, Iftee MA. Bangla natural language processing: A comprehensive review of classical machine learning and deep learning based methods. CoRR. 2021 May.

17. Kowsher M, Uddin MJ, Tahabilder A, Prottasha NJ, Ahmed M, Alam KR, Sultana T. BnVec: Towards the development of word embedding for Bangla language processing. Int. J. Eng. Technol. 2021;10(2):95.

18. Kowsher M, Tithi FS, Alam MA, Huda MN, Moheuddin MM, Rosul MG. Doly: Bengali chatbot for bengali education. In2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) 2019 May 3 (pp. 1-6). IEEE.

19. Bhowmik NR, Arifuzzaman M, Mondal MR, Islam MS. Bangla text sentiment analysis using supervised machine learning with extended lexicon dictionary. Natural Language Processing Research. 2021 Mar 22;1(3-4):34-45.

20. Kabir MK, Islam M, Kabir AN, Haque A, Rhaman MK. Detection of depression severity using Bengali social media posts on mental health: study using natural language processing techniques. JMIR Formative Research. 2022 Sep 28;6(9):e36118.

Appendix

1. Model Building

```
import nltk
# Sample sets of positive and negative words
positive_words = ["ভাল", "শুভ", "আনন্দ ", "সুখ", "সুন্দর", "চমৎকার", "দারুণ"]
negative_words = ["খারাপ", "দুঃ খ", "কষ্ট ", "বেদনা", "ভয়ংকর", "বাজে","ভয়াবহ","না","নয়","নি"]
# Function to perform sentiment analysis
def sentiment_analysis(text):
# Tokenize the text (split into words)
words = text.split()
# Initialize counters for positive and negative words
positive_count = 0
negative_count = 0
# Count occurrences of positive and negative words in the text
for word in words:
if word in positive_words:
positive_count += 1
elif word in negative_words:
negative_count += 1
# Determine the sentiment based on the counts
if positive_count > negative_count:
return "Positive Sentiment"
elif negative_count > positive_count:
return "Negative Sentiment"
else:
return "Neutral Sentiment"
# Test Bengali sentence
sentence = " আজকে খাবো না "
# Perform sentiment analysis on the sentence
result = sentiment_analysis(sentence)
#Tokenize
tokens = sentence.split()
# Print the result
print("Sentence:", sentence)
print(tokens)
print("Sentiment:", result)
```

Output:

```
Sentence: আজকে খাবো না
['আজকে', 'খাবো', 'না']
Sentiment: Negative Sentiment
```

2. Data Preparation

```
import pandas as pd
# Load predefined word lists from Excel files
def load_word_list(file_path, sheet_name):
try:
xls = pd.ExcelFile(file_path)
df = xls.parse(sheet_name)
if 'Words' in df.columns:
return df['Words'].tolist()
```

```
else:
raise KeyError(f"'Words' column not found in {file_path}")
except Exception as e:
print(f"Error loading word list from {file_path}: {e}")
return []
# Paths to Excel files containing word lists for each POS and sentiment
noun_file = 'C:/Users/Lenovo/Desktop/Noun words document.xlsx'
pronoun_file = 'C:/Users/Lenovo/Desktop/Pronoun words document.xlsx'
verb_file = 'C:/Users/Lenovo/Desktop/Verb words document.xlsx'
adjective_file = 'C:/Users/Lenovo/Desktop/Positive words document.xlsx'
adverb_file = 'C:/Users/Lenovo/Desktop/Adverb words document.xlsx'
conjunction_file = 'C:/Users/Lenovo/Desktop/Conjunction words document.xlsx'
interjection_file = 'C:/Users/Lenovo/Desktop/Interjection words document.xlsx'
positive_file = 'C:/Users/Lenovo/Desktop/Positive words document.xlsx'
negative_file = 'C:/Users/Lenovo/Desktop/Negative words document.xlsx'
neutral_file = 'C:/Users/Lenovo/Desktop/Neutral words document.xlsx'
# Load word lists from Excel files
nouns = load_word_list(noun_file, 'Sheet1')
pronouns = load_word_list(pronoun_file, 'Sheet1')
verbs = load_word_list(verb_file, 'Sheet1')
adjectives = load_word_list(adjective_file, 'Sheet1')
adverbs = load_word_list(adverb_file, 'Sheet1')
conjunctions = load_word_list(conjunction_file, 'Sheet1')
interjections = load_word_list(interjection_file, 'Sheet1')
positive_words = load_word_list(positive_file, 'Sheet1')
negative_words = load_word_list(negative_file, 'Sheet1')
neutral_words = load_word_list(neutral_file, 'Sheet1')
# Function to perform rule-based POS tagging
def get_pos_tag(word):
word = word.lower() # Convert to lowercase for comparison
if word in nouns:
return 'NN' # Noun
elif word in pronouns:
return 'PRP' # Pronoun
elif word in verbs:
return 'VB' # Verb
elif word in adjectives:
return 'JJ' # Adjective
elif word in adverbs:
return 'RB' # Adverb
elif word in conjunctions:
return 'CC' # Conjunction
elif word in interjections:
return 'UH' # Interjection
else:
return 'UND' # Undefined
# Function to perform rule-based sentiment analysis
def get_sentiment(sentence):
sentence = sentence.lower()
for word in positive_words:
if word in sentence:
return 'Positive'
for word in negative_words:
if word in sentence:
return 'Negative'
for word in neutral_words:
```

```
if word in sentence:
return 'Neutral'
return 'Neutral' # Default to neutral if no match
# Load your Excel file containing sentences
file_path = input ("Please provide the file path of the Excel file with sentences: ")
try:
xls = pd.ExcelFile(file_path)
df = xls.parse('Sheet1')
if 'Sentences' not in df.columns:
raise KeyError("The 'Sentences' column is missing from the input file.")
# Initialize lists for POS tagging and sentiment analysis
pos_tags = []
sentiments = []
# Process each sentence
for sentence in df['Sentences']:
words = sentence.split() # Split sentence into words
pos_tagged_sentence = [(word, get_pos_tag(word)) for word in words] # POS
pos_tags.append(pos_tagged_sentence)
sentiment = get_sentiment(sentence) # Sentiment analysis
sentiments.append(sentiment)
# Add the POS tagging and sentiment analysis results back to the dataframe
df['POS Tagging'] = pos_tags
df['Sentiment'] = sentiments
# Save the modified dataframe to a new Excel file
output_path = 'Processed_Sentiment_POS_Tags.xlsx'
df.to_excel(output_path, index=False)
print(f"POS tagging and sentiment analysis complete. File saved as {output_path}")
except Exception as e:
print(f"Error processing file: {e}")
```

Output:

```
Please provide the file path of the Excel file with sentences:
C:/Users/Lenovo/Desktop/Book.xlsx
POS tagging and sentiment analysis complete. File saved as
Processed_Sentiment_POS_Tags.xlsx
```

3. Model Evaluation

```
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
# Load the Excel file with the actual and predicted sentiment
file_path = 'Processed_Sentiment_POS_Tags.xlsx' # Update this path if needed
df = pd.read_excel(file_path)
# Ensure 'Actual Sentiment' and 'Sentiment' columns are present
if 'Actual Sentiment' not in df.columns or 'Sentiment' not in df.columns:
raise KeyError("The 'Actual Sentiment' or 'Sentiment' column is missing from the dataset")
```

```
# Extract actual and predicted sentiments
y_true = df['Actual Sentiment']
y_pred = df['Sentiment']
# 1. Calculate accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
# 2. Confusion Matrix
cm = confusion_matrix(y_true, y_pred, labels=['Positive', 'Negative', 'Neutral'])
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Positive', 'Negative', 'Neutral'])
plt.xlabel('Predicted Sentiment')
plt.ylabel('Actual Sentiment')
plt.title('Confusion Matrix')
plt.show()
# 3. Sentiment Distribution (Actual vs Predicted)
plt.figure(figsize=(8, 6))
actual_distribution = y_true.value_counts()
predicted_distribution = y_pred.value_counts()
# Bar plot comparing actual and predicted sentiment distributions
actual_distribution.plot(kind='bar', color='lightblue', label='Actual', alpha=0.7)
predicted_distribution.plot(kind='bar', color='orange', label='Predicted', alpha=0.
plt.title('Sentiment Distribution: Actual vs Predicted')
plt.xlabel('Sentiment')
plt.ylabel('Frequency')
plt.legend()
plt.show()
# 4. Classification Report (Precision, Recall, F1-Score)
report = classification_report(y_true, y_pred, labels=['Positive', 'Negative', 'Neutral'])
print("Classification Report:")
print(report)
# Optionally, plot precision, recall, F1-score as a bar chart
classification_metrics = classification_report(y_true, y_pred, labels=['Positive', 'Negative', 'Neutral'])
metrics_df = pd.DataFrame(classification_metrics).T.iloc[:-3, :3] # Only extract p
metrics_df.plot(kind='bar', figsize=(10, 6), colormap='viridis', alpha=0.85)
plt.title('Precision, Recall, and F1-Score for Each Sentiment Class')
plt.xlabel('Sentiment Class')
plt.ylabel('Score')
plt.xticks(rotation=0)
plt.show()
```

Output:

```
Accuracy: 93.94%
```

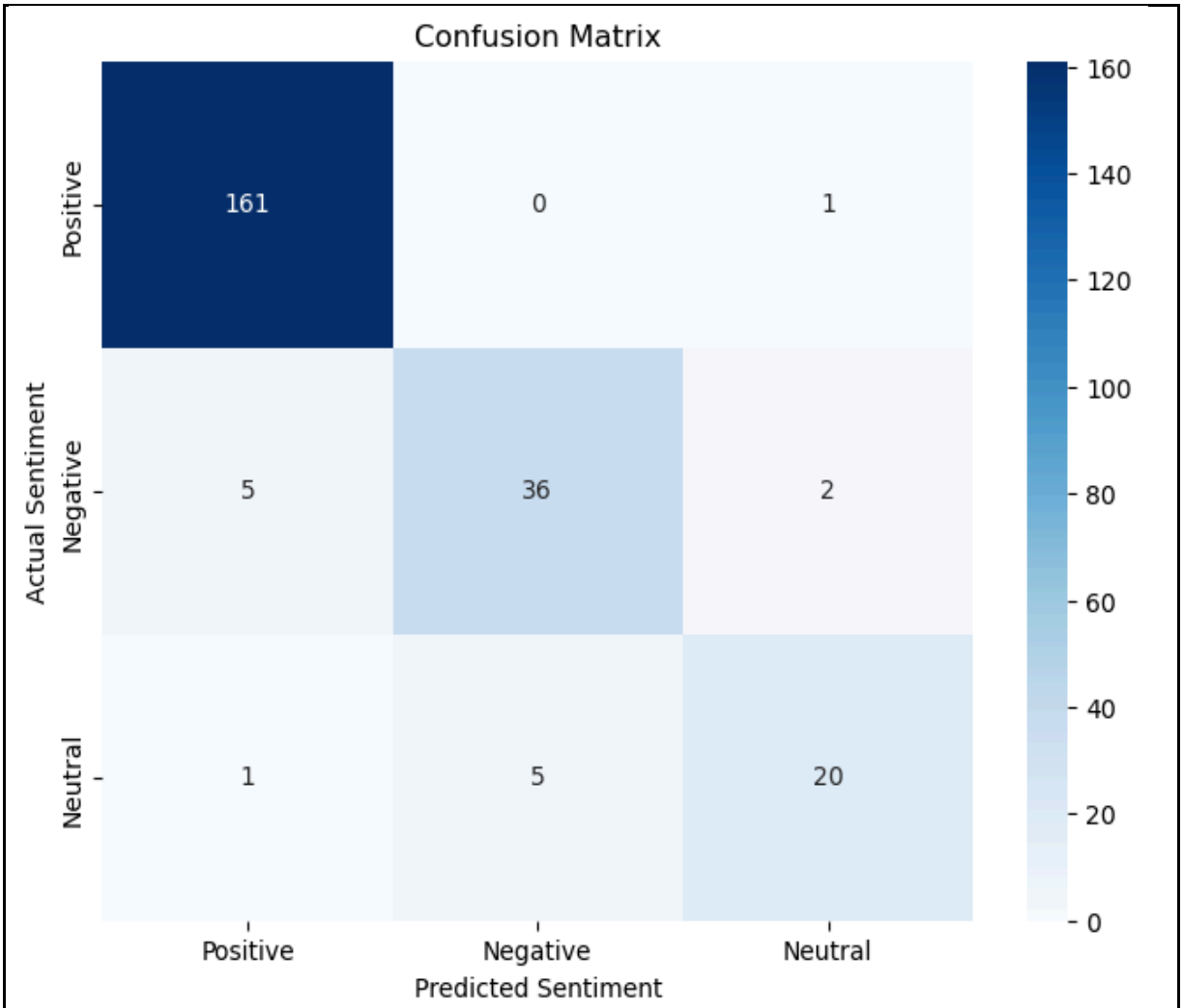

Fig 1-Confusion matrix

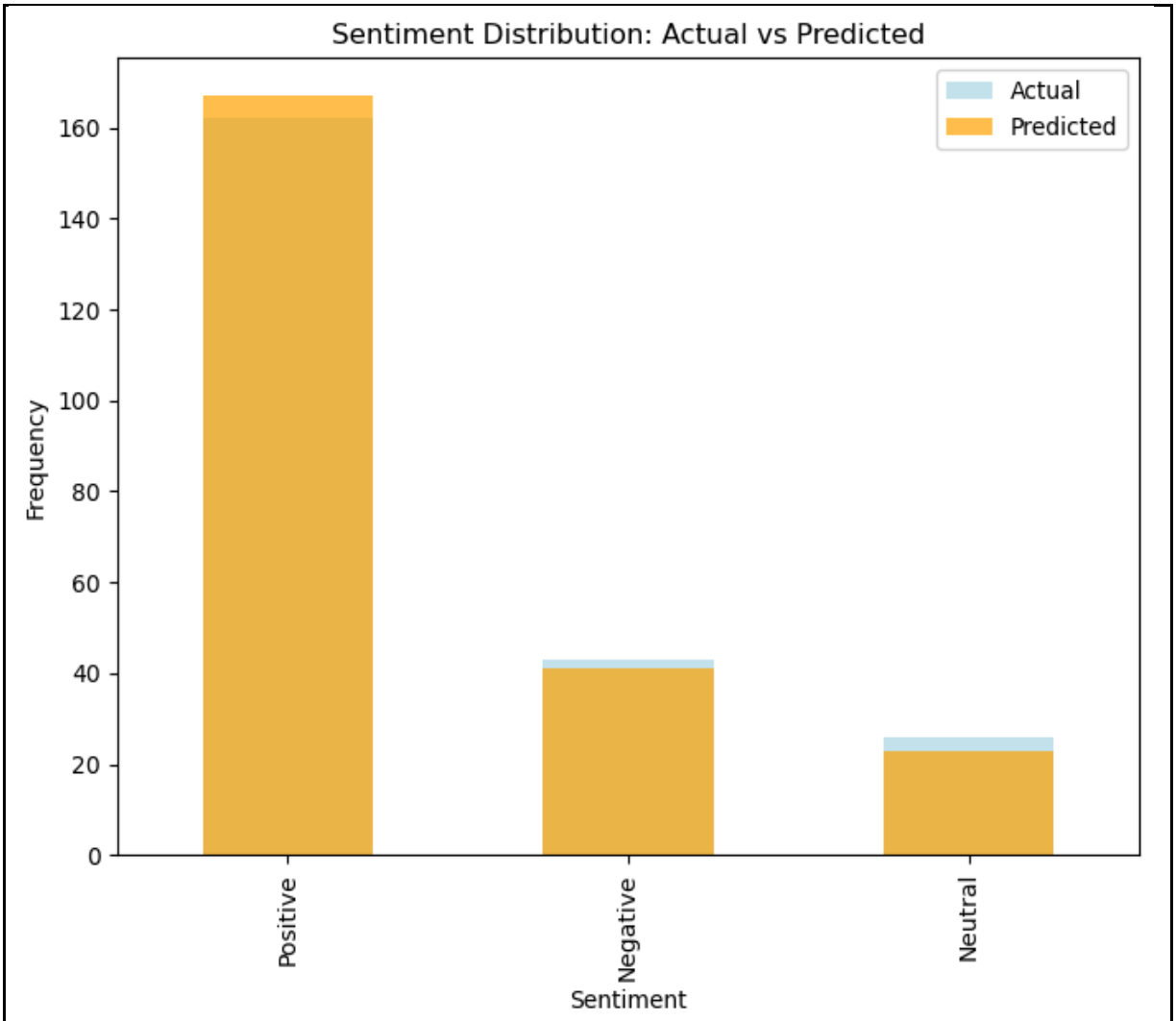

Fig 2- Sentiment distribution

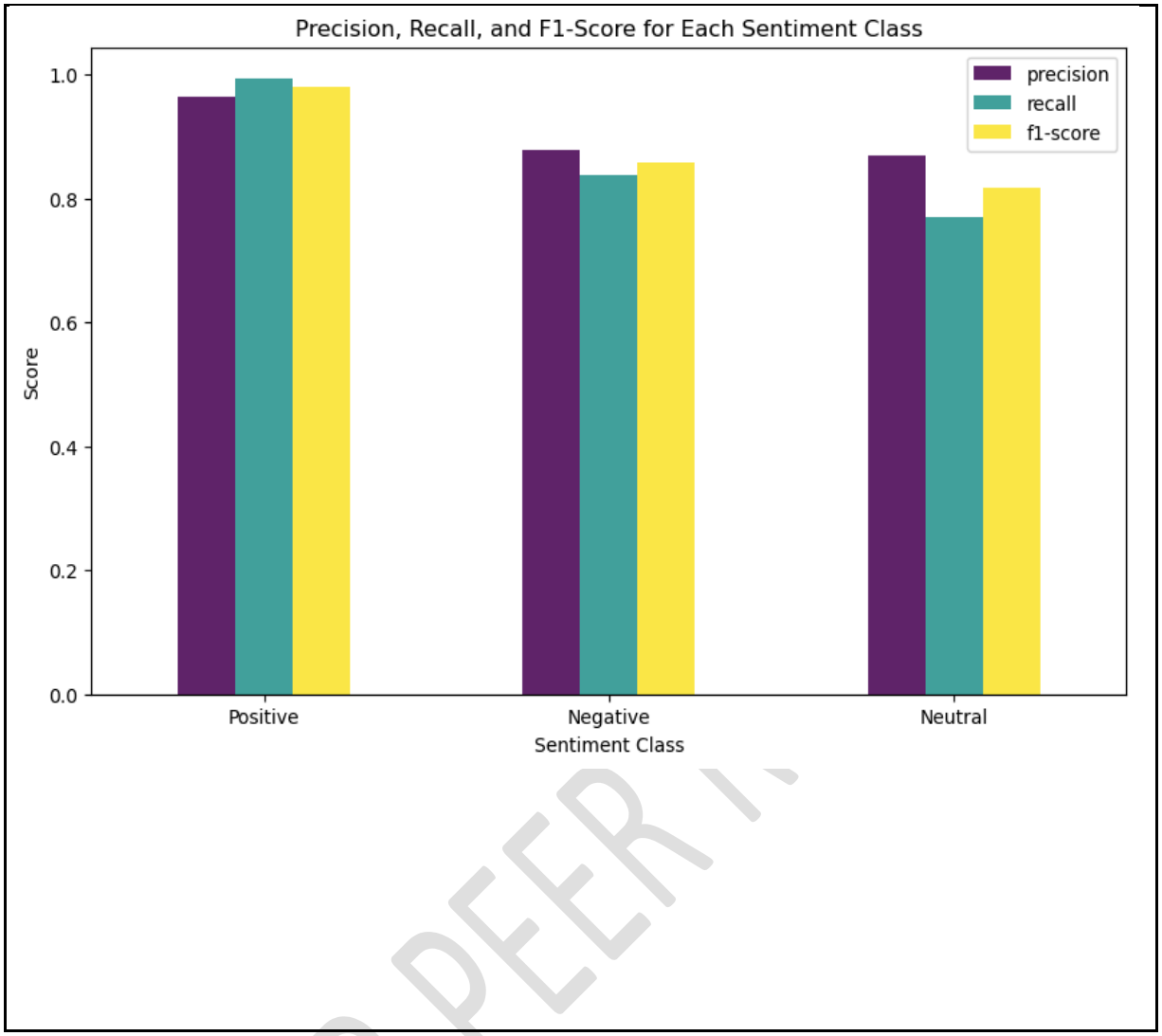

Fig 3- Precision, recall and F1 score of each sentiment vlass