



Enhancing NLP for Myanmar News Classification and Summarization

[KE-51129]

Aung Khant Myat [TNT -1000]

Tun Ye Minn [TNT -984]

Khin Nyo Nyo Theint [TNT - 970]

Submitted Date: 4/3/2024

A Project
Presented Enhancing NLP for Myanmar News Classification and
Summarization

A Term to
Faculty of Computer Science
University of Information Technology, Yangon

In Partial Fulfillment
Of the Requirement of the Degree
Bachelor of Computer Science

Submitted By
Aung Khant Myat [Tanata - 1000]
Tun Ye Minn [Tanata - 984]
Khin Nyo Nyo Theint [Tanata - 970]

[Final Year]

Acknowledgment

This project book has been accomplished with the help of our beloved teachers. We would like to deeply thank them for their extra assistance and support. We wish to express our sincere gratitude to our supervisors, Dr. Aung Nway Oo, Dr. Thinn Thinn Wai, Dr. Hmway Hmway Tar, Dr. Win Win Thant, and Dr. Thet Thet Zin, whose guidance and support have been instrumental in expanding our knowledge in our research domain. Throughout our final year project, they generously shared their expertise in Natural Language Processing, aiding us in overcoming various challenges. We are also thankful to the Faculty of Computer Science at the University of Information Technology for providing us with the opportunity and support to undertake this project.

Lastly, we extend our heartfelt appreciation to our friends and family for their unwavering support and encouragement. Their love and patience have been a constant source of strength, helping us persevere through difficulties. We are grateful to everyone who has contributed to our journey.

Abstract

Text classification focuses on a dataset comprising Myanmar news articles categorized into four distinct topics: Politics, Entertainment, Crime, and Business, with each category consisting of 180 unique articles, totaling 720 articles. These articles are then segmented into sentences for classification task. Preprocessing techniques involve regular expression-based cleaning and tokenization utilizing the Pyidaungsu library, followed by segmentation of articles into sentences using the Burmese character "။" and subsequent removal of stopwords. The dataset is then split into training and testing sets, with an 80-20 ratio, respectively. Experimentation involves employing the different vectorization techniques with an RBF Support Vector Machine (SVM) model. Results indicate that the TFID Vectorizer demonstrates superior performance, achieving an accuracy of 83%. Additionally, feature selection via the Chi Square Feature selector is applied to further optimize performance.

Summarization techniques can be categorized into two main approaches: extractive and abstractive summarization. Extractive summarization involves selecting a subset of sentences from the original text as the summary, whereas abstractive summarization generates the entire summary from scratch based on the input text. The human-like nature of abstractive summarization, where the summary is created a new, presents a challenge due to its complexity compared to extractive summarization. For extractive summarization, cosine similarity and the TextRank algorithm were employed. For abstractive summarization, a transformer-based seq2seq model, specifically the mT-5 model, was utilized, leveraging the XL-Sum dataset. As a result, the extractive summarization of the proposed system yields 43.14 of the ROUGE-1 score, while the abstractive summarization contributes 36.67 of the ROUGE-1 score.

Table of Contents

Acknowledgment	iii
Abstract	iv
Table of Contents	v
LIST OF FIGURES	viii
LIST OF TABLES	x
Chapter 1 Introduction	1
1.1 Problem Statement	2
1.2 Motivation	2
1.3 Objectives of the Project	3
1.4 Contributions	4
1.5 Organizations of Project Book	4
1.6 Related Works	5
Chapter 2 THEORY BACKGROUND	7
2.1 Natural Language Processing.....	7
2.2 Myanmar Language.....	7
2.3 Myanmar Word Segmentation	9
2.4 Vectorization	10
2.4.1 TF-IDF Vectorizer.....	10
2.4.2 Count Vectorizer	11
2.4.3 Word2Vec Vectorizer	12
2.5 Text Mining and Automatic Text Classification.....	13
2.5.1 Multinomial Naive Bayes.....	14
2.5.2 Support Vector Machine (SVM)	15
2.5.3 Random Forest.....	16
2.6 Summarization	16
2.6.1 Extractive Summarization	17
2.6.2 Abstractive Summarization	18
Chapter 3 Methodology	20
3.1 Myanmar News Classification	20
3.1.1 System Architecture	20
3.1.2 Data Collection	21

3.1.3 Data Preprocessing	22
3.2 Myanmar News Summarization.....	29
3.2.1 Extractive Summarization	29
3.2.2 Abstractive Summarization	34
Chapter 4 Implementation.....	38
4.1 Tools and Technologies used	38
4.2 Myanmar News Classification	39
4.2.1 Data.....	39
4.2.2 Data Preprocessing	40
4.2.3 Splitting Dataset	43
4.2.4 Application of Vectorization	43
4.2.5 Application of Chi-Square.....	44
4.2.6 Models Applied for Topic Classification.....	45
4.2.6.1 Multinomial Naive Bayes (MNB)	46
4.2.6.2 Support Vector Machine (SVM)	46
4.2.7 Grid Searching for SVM.....	48
4.3 Extractive Myanmar News Summarization	50
4.3.1 Data Preprocessing	50
4.3.2 Vectorization	51
4.3.3 Cosine Similarity Scores	52
4.3.4 Sentence Selection.....	52
4.4 Abstractive Myanmar News Summarization	53
4.4.1 Data Implementation	53
4.4.2 Implementation of Tokenization	53
4.4.3 Optimization of Model Training.....	54
4.4.4 Implementation of ROUGE Score evaluation	55
4.4.5 Implementation of Data Loader.....	56
4.4.6 Finetuning using Seq2SeqTrainer Class.....	57
Chapter 5 Evaluation of Experimental Results.....	59
5.1 Myanmar News Classification	59
5.1.1 Performance Evaluation of Naive Bayes Classifier	60
5.1.2 Performance Evaluation of SVM Classifier	60

5.1.3 Performance Evaluation of Random Forest Classifier	61
5.1.4 Comparison of the Accuracies with Various Vectorization Methods	61
5.1.5 Comparison of the Accuracies with Various ML Methods	62
5.2 Myanmar News Summarization.....	63
5.2.1 Performance Evaluation of Extractive Myanmar News Summarization.....	65
5.2.1 Performance Comparison of Multilingual Model and the proposed Monolingual Model of Abstractive Summarization on XL-Sum Dataset	65
5.2.2 Performance Comparison of Low-Resource mT5 Model and the proposed Monolingual Model of Abstractive Summarization on XL-Sum Dataset	66
Chapter 6 Application Implementation.....	68
6.1 Streamlit Cloud	68
6.2 Myanmar News Classification	69
6.3 Myanmar News Summarization.....	70
Chapter 7 Conclusion.....	73
7.1 Conclusion.....	73
7.2 Future Work	74
REFERENCES	76
Appendix.....	78

LIST OF FIGURES

Figure 2.1 Hierarchy of Myanmar Grammatical Units and Constructions.....	8
Figure 2.2 Word2Vec relationships between words	13
Figure 2.3 SVM Classifier	15
Figure 2.4 Random Forest Classifier	16
Figure 2.5 Example of seq2seq encoder-decoder architecture	19
Figure 3.1 Proposed System Architecture for Myanmar News Classification	20
Figure 3.2 The Myanmar News Dataset	22
Figure 3.3 Vectorizer Features on Business Category.....	24
Figure 3.4 Vectorizer Features on Crime Category	24
Figure 3.5 Vectorizer Features on Crime Category	24
Figure 3.6 Vectorizer Features on Crime Category	25
Figure 3.7 Finding the best numbers of features using Chi-Square.....	26
Figure 3. 8 Proposed System Architecture for Extractive Myanmar News Summarization	30
Figure 3.9 Proposed System Architecture for Abstractive Myanmar News Summarization	34
Figure 3.10 Languages covered by the XL-Sum Dataset	35
Figure 4.1 Myanmar News Articles.....	40
Figure 4.2 Sentence Segmentation.....	41
Figure 4.3 Text Cleaning	41
Figure 4.4 Word Segmentation	42
Figure 4.5 Stopword Removal	42
Figure 4.6 Splitting Dataset	43
Figure 4.7 Application of TF-IDF Vectorizer.....	44
Figure 4.8 Searching k_best using SVM model.	45
Figure 4.9 Application of Multinomial Naive Bayes (MNB).....	46
Figure 4.10 Application of RBF Support Vector Machine (SVM)	47
Figure 4.11 Application of Random Forest (RF).....	48
Figure 4.12 Application of Grid Search for SVM	49

Figure 4.13 Data preprocessing for Extractive Summarization.....	51
Figure 4.14 Vectorization for Extractive Summarization.....	51
Figure 4.15 Computing Cosine Similarity Scores for Extractive Summarization.....	52
Figure 4.16 Sentence Selection for Extractive Summarization	52
Figure 4.17 XL-Sum data installation for Abstractive Summarization	53
Figure 4.18 loading tokenizer in pre-trained google/mt5-small model.	53
Figure 4.19 Applying tokenization for dataset	54
Figure 4.20 Configuring the model.....	55
Figure 4.21 Applying Data Collator	55
Figure 4.22 prepare metrics function for ROUGE Score evaluation.....	56
Figure 4.23 Implementation of DataLoader.....	57
Figure 4.24 Implementation of Seq2Seq Trainer Class	58
Figure 5.1 Comparison of the Accuracies with Various Models for Classification	63
Figure 5.2 Comparison of the Accuracies with Various Models for Summarization.....	67
Figure 6.1 Streamlit Cloud Process Flow	68
Figure 6.2 Myanmar News SVM Classifier's Interface	69
Figure 6.3 Myanmar News Naive Bayes Classifier's Interface.....	70
Figure 6.4 Evaluation Interface of Myanmar News Classification.....	70
Figure 6.5 Extractive Myanmar News Summarization's Interface	71
Figure 6.6 Monolingual Abstractive Myanmar News Summarization's Interface.....	72
Figure 6.7 Multilingual Abstractive Myanmar News Summarization's Interface	72

LIST OF TABLES

Table 2.1 General Structure of Myanmar Text Sentence	8
Table 3.1 Description of Dataset Features	21
Table 3.2 Sample Input Article	32
Table 3.3 Ranked Sentences with their Cosine Similarity Scores between Sentence and PageRank Algorithm.....	33
Table 4.1 Python Libraries Version	38
Table 4.2 Google Collab Specifications	38
Table 4.3 Myanmar News Sentences.....	40
Table 5.1 Experimental Results for Naive Bayes Classifier.....	60
Table 5.2 Experimental Results for SVM Classifier	60
Table 5.3 Experimental Results for Random Forest Classifier.....	61
Table 5.4 Compare Accuracies with Various Vectorizers.....	61
Table 5.5 Accuracies of Different Models with Myanmar News Dataset.....	62
Table 5.6 Experimental Results for Extractive Summarization.....	65
Table 5.7 Accuracies of multilingual model and monolingual model with XL-Sum Dataset.....	66
Table 5.8 Accuracies of low-resource model and monolingual model with XL-Sum Dataset	66

Chapter 1 Introduction

With the rapid growth of the internet, the availability of online text information has been considerably increased. As a result, text mining has become the key technique for handling and organizing text data and extracting relevant information from massive amounts of text data. Text mining may be defined as the process of analyzing text to extract information from it for purposes, for example, information extraction and information retrieval. Typical text mining tasks include text classification, text summarization, text clustering, entity extraction, and sentiment analysis.

Text classification involves many applications like text filtering, document organization, classification of news stories, searching for interesting information on the web, spam e-mail filtering, etc. These are language-specific systems mostly designed for English, European, and other Asian languages but very little work has been done for the Myanmar language. Therefore, developing classification systems for Myanmar documents is a challenging task due to morphological richness and scarcity of resources of the language like automatic tools for tokenization, feature selection, and stemming, etc.

In this system, a text classification system for Myanmar news is proposed. The proposed system is implemented by using a supervised learning approach which is defined as assigning the pre-defined category labels to the text documents based on the likelihood suggested by the training set of labelled documents. For the training data set, news from Myanmar media websites is manually collected, labeled, and stored in the training data set. The chi-square function is used as a feature selection method and a classification algorithm is applied in implementing the text classifier.

This system used Random Forest, Naïve Bayes, and SVM Classifier in Myanmar text classification. This system aims at making a comparative study between the mentioned algorithms on a Myanmar data set to give out beset News Topic Classification.

Automatic text summarization holds significant promise across various domains in today's technological landscape, particularly in condensing content like news articles and research papers. A lot of work has already been done in summarizing English language text. But very little work is being done in summarizing Myanmar Languages.

Summarization techniques can be categorized into two main approaches: extractive and abstractive summarization. Extractive summarization involves selecting a subset of sentences from the original text as the summary, whereas abstractive summarization generates the entire summary from scratch based on the input text. The human-like nature of abstractive summarization, where the summary is created anew, presents a challenge due to its complexity compared to extractive summarization.

In this study, the aim is to conduct both extractive and abstractive summarization on Myanmar news articles. For extractive summarization, cosine similarity and the Text Rank algorithm were employed. For abstractive summarization, a transformer-based seq2seq model, specifically the mT-5 model, was utilized, leveraging the XL-Sum dataset.

1.1 Problem Statement

Unlike most of the popular languages, our Myanmar language, with its rich cultural and linguistic diversity, has a distinctive set of challenges for NLP tasks. In the case of Topic classification, implementing robust Topic Classification models can provide a structured framework for understanding and organizing the diverse range of topics. Sadly, most of these tasks haven't been done related to Myanmar language widely. Developing such tasks can help in content organization and to facilitate more targeted information retrieval for Myanmar News media and researchers.

As for summarization, the Myanmar texts, which are often lengthy and intricate, benefit greatly from Text Summarization. Generating concise and coherent summaries not only facilitates quicker comprehension but also addresses the challenge of information overload. In domains like news articles, legal documents, and academic research, automatic summarization can significantly improve accessibility and information retrieval.

1.2 Motivation

In recent years, Natural Language Processing (NLP) has emerged as a vital technology in various fields, revolutionizing how to interact with and understand textual data. However, despite

significant advancements in NLP, many languages, including Myanmar, still lack comprehensive NLP solutions tailored to their unique linguistic characteristics and societal needs.

The Myanmar language, with its complex script and rich linguistic nuances, presents a challenging yet rewarding frontier for NLP research and development. Particularly in the domain of news analysis, there exists a pressing need for effective tools that can enhance the accessibility, comprehension, and dissemination of information within the Myanmar-speaking community.

By focusing on classification and summarization techniques tailored specifically for Myanmar news content, this system aims to address these pressing challenges and contribute to the advancement of NLP in the Myanmar language domain. Through the development of tailored algorithms and methodologies, the system seeks to empower users with the ability to efficiently categorize news articles and generate concise summaries, thereby facilitating easier information consumption and knowledge dissemination.

Furthermore, by bridging the gap between cutting-edge NLP technologies and the linguistic intricacies of the Myanmar language, this research endeavors to foster greater inclusivity and accessibility in the digital information landscape. By enabling more effective news analysis and comprehension, the system aspires to empower individuals, communities, and policymakers to make well-informed decisions.

1.3 Objectives of the Project

The primary goal of this research is to provide a better Myanmar Topic Classification and Summarization system to the users. The Topic Classification task offer four domain classification of news contents or articles. The domain are Crime, Business, Politic and Entertainment. The system will be able to classify most of the up to date or old news into one of four categories. In the case of Topic Summarization, there is actually three summarization techniques such as Extractive Summarization, Mono Abstractive Summarization and Muli Abstractive Summarization. In the context of Extractive Summarization, it offers summarization of different levels, meaning users can generate up to desired number of sentences. On the other hand, abstractive offers monolingual summarization and multilingual summarization. Meaning they generate summarization base on type of mono and multi-NLP models applied. Depending on the two different models usage, an

input given can be summarized into two different outputs. The system also aims to contribute methodologies to Myanmar NLP research, particularly in the domain of news analysis and language-specific NLP applications to advance the state-of-the-art NLP for Myanmar language processing.

1.4 Contributions

For the development of this system, our team has contributed unique techniques to deliver to acceptable output or to optimize the performance of the system. In the case of Topic Classification, we built an unique Myanmar News Article Corpus for research and applied suitable stop words to filter the corpus sentence words for further processing. Later, we applied the best vectorization technique (TF-IDF) by many hyperparameter tunings. Depending on the vectorized features generated, we selected only the important features for the Topic Classification using feature selection methods. Moreover, for the optimal model development, applied grid searching for finding the best model to be trained on. In the case of summarization, we developed a comprehensive framework for both extractive and abstractive text summarization methods. Then, we evaluated the performance of the summarization methods, comparing their effectiveness in generating high-quality summaries. Additionally, to improve the accuracy and coherence of abstractive summarization, trained the mT-5 model on Myanmar language data. Our team would like to contribute the development of effective text summarization systems capable of handling diverse linguistic contexts and producing high-quality summaries.

1.5 Organizations of Project Book

This project is organized as follows:

Chapter 1 presents the introduction of the whole project.

Chapter 2 presents the background theory applied to the whole project.

Chapter 3 presents the methodology of the whole project.

Chapter 4 presents the implementation of the whole project.

Chapter 5 presents the results and analysis of the project.

Chapter 6 presents the Application Implementation of the project.

Chapter 7 presents the conclusion of the project.

1.6 Related Works

Many algorithms have been applied for Automatic Text Classification. Most studies have been devoted to English and other Latin languages. However, very few research have been carried out on Myanmar text.

For the Myanmar Language, In the paper [1], T.T.Zaw and K.M.Soe proposed the Myanmar language text classification using TF-IDF Genetic Algorithm Classifier. Term frequency-inverse document frequency (tf_idf) algorithm was used to select related features according to their labeled document. The experimental results showed that their proposed system worked as a successful classifier with a 90% accuracy rate. K.T.Nwet, and A.H.Khine from the University of Computer Studies, Yangon have developed an Automatic Myanmar News Classifier using supervised Methods. The chi-square function was used as a feature selection method and applied Classification Algorithms were Naïve Bayes and k-Nearest Neighbors (KNN). They developed a Myanmar News corpus which consists of 1200 documents belonging to 4 categories. Results showed that the K-NN classifier is better than the naïve Bayesian classifier. However, they concluded that the problem with K-NN is that its time complexity is high but has better accuracy than others [2].

Many supervised learning algorithms have been applied to the area of text classification, using pre-classified training document sets. Those algorithms, that used classification, include K-Nearest Neighbors (K-NN) classifier, Naïve Bayes (NB), decision trees, the Random Forest, Support Vector Machines (SVM), and Neural Networks.

According to S.S.Lwin and K.T.Nwet, in their paper titled "Myanmar news summarization using different word representations," the authors proposed a centroid-based word embedding summarizer for Myanmar news, employing various word embedding models to compare results with bag-of-words summarization. Experiments utilize local and international news datasets, with

cosine similarity used to rank sentences based on their similarity to the centroid vector. Results indicate that models like FastText and BPEmb outperform the baseline BOW model, with the BPEmb-based centroid summarizer achieving the highest ROUGE scores under specific conditions. These findings highlight the superiority of word embedding methods over BOW models, emphasizing the significance of capturing semantic and syntactic relationships for enhanced summarization outcomes [3].

Y.Thu and W.P.Pa explore text summarization techniques specifically tailored towards Myanmar news articles. The authors propose a Recursive RNN model for headline prediction, which uses an encoder-decoder LSTM model to generate a single word forecast and calls it recursively to create text summarizations in the form of news headlines. They collect 5,000 Myanmar news articles to train the headline prediction model and evaluate its performance relative to sequence-to-sequence models using ROUGE score values. Their findings indicate that the Recursive RNN model significantly outperforms other models, demonstrating its potential for accurately and concisely generating headlines for Myanmar news articles [4].

Current scholarly works primarily concentrate on extractive summarization techniques, highlighting the need for further research and innovation in the domain of abstractive text summarization utilizing transformer Seq2Seq models for Myanmar language processing.

Chapter 2 THEORY BACKGROUND

2.1 Natural Language Processing

Natural language processing is called understanding and learning from text data. It is the field of artificial intelligence (AI) focused on enabling computers to comprehend and interpret human language from text data. Human language is complex and diverse, making developing NLP applications challenging. NLP faces various challenges, in dealing with unstructured data generated from online sources and social media. Understanding the semantic meaning of words is another obstacle, as many words share similar meanings across languages, requiring machines to grasp the context. The same word has different meanings, and understanding different meanings of the same spelling is one of the challenges in NLP. Two primary methods used in natural language processing are syntax and semantic analysis. Syntax involves organizing words in a sentence to convey grammatical meaning, while semantics focuses on understanding word usage and meaning. Syntax techniques include parsing (analyzing sentence structure), word segmentation (dividing text into units), sentence breaking (identifying sentence boundaries), morphological segmentation (grouping words), and stemming. For semantic analysis, techniques such as word sense disambiguation, named entity recognition, and natural language generation are utilized in NLP.

Text classification and text summarization are challenging tasks in both Natural Language Processing (NLP) and Machine Learning (ML). Extracting key information from diverse textual sources, including social media, IoT devices, and mobile communications, is essential for both tasks. Semantic and context understanding is crucial for accurate classification and effective summarization.

2.2 Myanmar Language

The Myanmar Language is the official Language of the Republic of the Union of Myanmar. The Myanmar alphabet consists of 33 letters and 12 vowels and is circular. It is a free-word-order and verb-final language, which usually follows the subject-object-verb (SOV) order. Text direction

in Myanmar is written horizontally, left to right. The Myanmar language is classified into two categories: formal and colloquial. Formal is used in literature, official work, and radio broadcasting. Colloquial is used in daily conversations. Myanmar has nine parts of speech: noun, pronoun, verb, adjective, adverb, particle, conjunction, post-positional marker, and interjection. Sentences are delimited by sentence boundary markers in the Myanmar language, but words are not always delimited by spaces. There is no definite rule for space insertion between words; sometimes spaces are placed between phrases. Word tokenization is a challenging task in the Myanmar language.

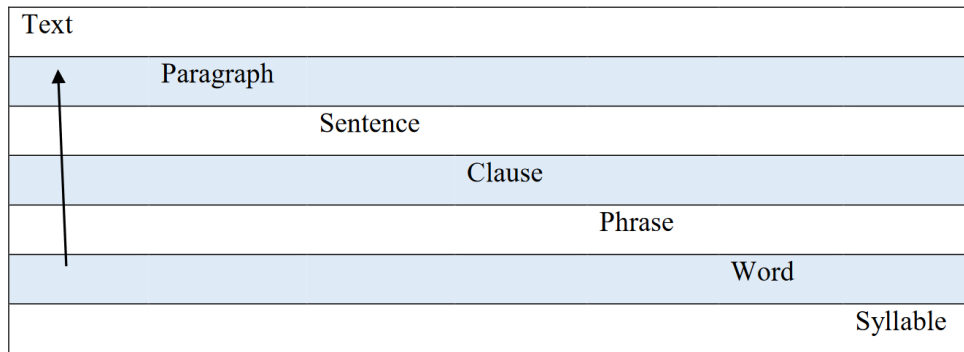


Figure 2.1 Hierarchy of Myanmar Grammatical Units and Constructions

The figure shows the basic hierarchy of Myanmar grammatical units and constructions. This hierarchy is the heuristic principle to establish a basic understanding of grammatical units and constructions in Myanmar. In the Myanmar language, a sentence is composed of two or more phrases, such as a noun phrase, and a verb phrase. A phrase consists of two or more words and Myanmar words are made up of one or more syllables. Table 2.1 shows the general structure of a Myanmar text sentence.

Table 2.1 General Structure of Myanmar Text Sentence

Sentence	ကျွန်တော်သည် လှပသော မိန်းကလေးကို ချစ်ပါသည်
Phrase	ကျွန်တော်သည်_လှပသော_မိန်းကလေးကို_ချစ်ပါသည်
Word	ကျွန်တော်_သည်_လှပသော_မိန်းကလေး_ကို_ချစ်_ပါ_သည်
Syllable	ကျွန်-တော်-သည်-လှ-ပ-သော-မိန်း-က-လေး-ကို-ချစ်-ပါ-သည်

Unlike English, Myanmar text does not use white space to separate words or syllables. It is also important for every language because it is the fundamental step for linguistic processing. It is necessary analysis of high-level language such as Text classification, Text summarization, and machine translation.

2.3 Myanmar Word Segmentation

In many NLP applications including machine translation, information retrieval, text classification, text summarization, etc., word segmentation is a pre-processing phase. There is no explicit word boundary in the Myanmar language. It can be written in sequence from left to right without space between words and sometimes there is space between phrases. Therefore, Myanmar word segmentation is a challenging task. Myanmar words are divided into simple words, compound words, and complex words. These are some examples of loanwords and compound words.

Compound Words

- ([sa]စ-letter + [Tai']တိုက်-building)= စတိုက် -post office
- (ရိုက်-hit+ နှိပ်-press)= နှိပ် -imprint
- ([awa']အဝတ်-cloth+[sh']လျှော် -wash+[ses']စက်-machine)= အဝတ်လျှော်စက် -washing machine

Loan words

- ကွန်ပျူတာ [kun pju ta] computer
- ကား (car)
- ပူတင်း:(pudding)

To get specific words from the Myanmar sentence, much research has been carried out in many ways including both supervised and unsupervised learning. The Myanmar Word Segmentation based on conditional random field (CRF) prediction is used in the proposed system.

2.4 Vectorization

Vectorization is a fundamental concept in natural language processing (NLP) and machine learning, particularly when working with text data. Vectorization involves the process of converting text data into a numerical format that can be used by machine learning algorithms. This transformation allows the algorithms to process and analyze the text data effectively. There are several techniques for vectorizing text data, with the most common ones being bag-of-words, TF-IDF (Term Frequency-Inverse Document Frequency), and word embeddings.

Vectorization is crucial for NLP tasks such as text classification, sentiment analysis, and information retrieval, as it enables the algorithms to process and understand the textual data. By representing text data as numerical vectors, machine learning models can effectively learn patterns and relationships within the data, ultimately improving the accuracy of the classification tasks.

2.4.1 TF-IDF Vectorizer

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique used to represent text data numerically, aiding in its processing and understanding by machine learning algorithms. It consists of two main components: Term Frequency (TF) and Inverse Document Frequency (IDF). TF measures how often a word appears in a document, while IDF reduces the weight of words that appear in many documents in the corpus. This helps address the issue of common words having high frequencies across documents. The TF-IDF score for a word in a document is calculated by multiplying its TF and IDF.

Term Frequency (TF):

Term Frequency (TF) is calculated as the number of times a term t appears in a document d divided by the total number of terms in the document.

Mathematically, TF is represented as

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

Inverse Document Frequency (IDF):

Inverse Document Frequency (IDF) is calculated as the logarithm of the total number of documents in the corpus divided by the number of documents containing the term, plus one to avoid division by zero for terms not present in the corpus. IDF is designed to give higher weights to terms that are rare across the corpus, making them more discriminative.

Mathematically, IDF is represented as

$$\text{IDF}(t) = \log \left(\frac{\text{Total number of documents in the corpus}}{\text{Number of documents containing term } t} + 1 \right)$$

TF-IDF Calculation:

Multiplying the TF and IDF results in a weighted score that represents the importance of a term within a document relative to its importance in the corpus. Terms with higher TF-IDF scores are considered more important vectors in a document.

Mathematically, TF-IDF is represented as

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

These vectors can then be used for various applications, such as measuring document similarity and ranking search results based on relevance, keyword extraction, and text analysis in various NLP applications.

2.4.2 Count Vectorizer

Count Vectorizer is a fundamental technique used in natural language processing (NLP) to convert text data into numerical vectors. Countvectorizer builds a vocabulary of all unique terms present in the corpus. It operates on the principle of creating a matrix where rows represent documents and columns represent terms (words or n-grams). Each cell in this matrix contains the count of how many times a particular term occurs in a particular document.

Mathematically, if denote:

- N as the total number of documents in the corpus,
- M as the total number of unique terms in the vocabulary, and
- $\text{count}(i, j)$ as the count of term j in document i ,

then the Countvectorizer matrix X can be represented as

$$X = \begin{bmatrix} \text{count}(1, 1) & \text{count}(1, 2) & \cdots & \text{count}(1, M) \\ \text{count}(2, 1) & \text{count}(2, 2) & \cdots & \text{count}(2, M) \\ \vdots & \vdots & \ddots & \vdots \\ \text{count}(N, 1) & \text{count}(N, 2) & \cdots & \text{count}(N, M) \end{bmatrix}$$

This matrix is typically sparse because most documents contain only a subset of the total vocabulary. Each document is thus represented as a high-dimensional vector, where each dimension corresponds to a term and the value represents the frequency of that term in the document.

Before constructing the Count Vectorizer matrix, text data is typically preprocessed to remove punctuation, convert text to lowercase, and perform tokenization. Optionally, stop words (commonly occurring words like "I", "II", "I") may be removed to improve the quality of the vector representations.

Countvectorizer is widely used in various text-based machine learning tasks such as text classification, clustering, and information retrieval. It provides a simple yet effective way to represent text data in a format suitable for machine learning algorithms.

2.4.3 Word2Vec Vectorizer

Word2Vec is a natural language processing technique used to obtain vector representations of words. These vectors capture information about the meaning of the word and their usage in context. Word2Vec represents each distinct word with a particular list of numbers called a vector.

The vectors are chosen carefully such that they capture the semantic and syntactic qualities of words.

After training, each word in the vocabulary is represented as a dense vector (word embedding) in the continuous vector space. These word embeddings capture semantic relationships between words, such as similarity and analogy. Words with similar meanings or appearing in similar contexts are represented by vectors that are closer together in the vector space.

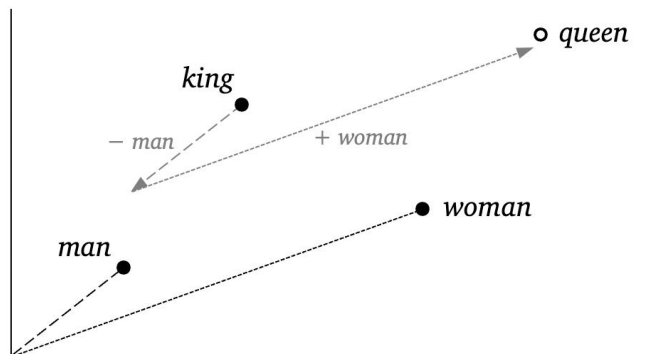


Figure 2.2 Word2Vec relationships between words

Word2Vec embeddings are widely used in various NLP tasks such as sentiment analysis, named entity recognition, machine translation, and document clustering (Topic Classification). They provide dense and continuous representations of words that capture semantic information, enabling better performance in downstream tasks compared to traditional sparse representations like one-hot encoding or count-based methods.

2.5 Text Mining and Automatic Text Classification

Text mining is the process of extracting valuable information from unstructured text documents, a part of the larger field of data mining. Text mining is a multidisciplinary field, where it involves techniques from other areas like information retrieval, text analysis, clustering, classification, categorization, summarization, machine learning, and data mining.

Text classification is an instance of text mining. The text classification process includes preprocessing the documents, extracting relevant features according to the feature in the training corpus, and applying a classification algorithm to classify documents into predefined categories.

Text classification classifies document d_j from the entire collection of document D and classifies it into one of the categories from $\{c_1, c_2, c_3, \dots, c_i\}$.

Text classification tasks can be divided into three sorts:

- 1) handwritten rule-based classifiers
- 2) supervised document classification where some external mechanism (such as human feedback) provides information on the correct classification for documents,
- 3) unsupervised document classification (also known as document clustering), where the classification must be done entirely without reference to external information.

The proposed system applied a supervised classification method to categorize Myanmar Language news from news websites into four categories -politics, business, entertainment, and crime. The supervised models used for Myanmar news classification include Naive Bayes, Support Vector Machines (SVM), and Random Forest.

2.5.1 Multinomial Naive Bayes

One of the methods used for classifying documents is Multinomial Naive Bayes (MNB), a probabilistic classification algorithm. It works on the principles of Bayes' theorem and is particularly suitable for problems involving text data with discrete features, such as word counts. The basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given in a document. In Naïve Bayes classifier, each document is viewed as a collection of words and the order of words is considered irrelevant. Given a document d for classification, the probability of each category c is calculated as follows:

$$P(c|d) = \frac{P(c) \prod_{w \in d} P(w|c)^{n_{wd}}}{P(d)}$$

where n_{wd} is the number of times word w occurs in document d , $P(w|c)$ is the probability of observing word w given class c , $P(c)$ is the prior probability of class c , and $P(d)$ is a constant that makes the probabilities for the different classes sum to one. $P(c)$ is estimated by the proportion of training documents pertaining to class c and $P(w|c)$ is estimated as

$$P(w|c) = \frac{1 + \sum_{d \in D_c} n_{wd}}{k + \sum_{w'} \sum_{d \in D_c} n_{w'd}}$$

where D_c is the collection of all training documents in class c , and k is the size of the vocabulary (i.e. the number of distinct feature words in all training documents). The additional one in the numerator is the Laplace correction, which corresponds to initializing each word count to one instead of zero. It requires the addition of k in the denominator to obtain a probability distribution that sums to one. This kind of correction is necessary because of the zero-frequency problem.

2.5.2 Support Vector Machine (SVM)

Support Vector Machines (SVM) is a machine learning technique applicable for classifying textual data, with diverse uses in areas like credit risk assessment, medical diagnosis, text classification, and information extraction. SVM is a large-margin classifier that finds a decision boundary between classes of data to separate them. In the case of text classification, SVM can be used to determine the best decision boundary between vectors that belong to a given category and vectors that do not. The algorithm works by encoding text data into vectors that the model can understand and then finding a hyperplane that creates a boundary between the classes of data. Notably, SVMs are particularly well-suited for handling high-dimensional data, as their classifier complexity correlates with the number of support vectors rather than the data dimensions. Moreover, SVMs consistently produce identical hyperplanes across repeated training sets and offer improved generalization capabilities. Even when dealing with sparse data, SVMs maintain their performance accuracy. SVMs tend to use an over-fitting protection mechanism that is independent of the dimensionality of the feature space.

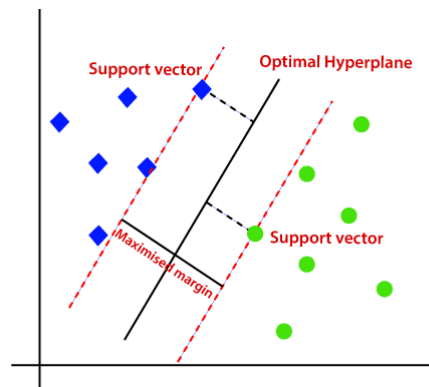


Figure 2.3 SVM Classifier

2.5.3 Random Forest

The Random Forest, or Random Decision Forest, is a supervised machine learning algorithm used for classification, regression, and other tasks by employing an ensemble of decision trees to make predictions. Random Forest Classification is an ensemble learning technique that significantly enhances classification accuracy and robustness. During training, it constructs multiple decision trees, with the final prediction being the most common class among these trees. Each decision tree in the random forest is built using a subset of the training data and a random subset of features. This diversity enhances model robustness while reducing the risk of overfitting. By utilizing bagging techniques like Bootstrap Aggregating, the algorithm creates diverse subsets. During training, each tree recursively partitions data based on features, selecting the best feature for splitting at each step to optimize information gain or Gini impurity. This recursive process continues until a predefined stopping criterion is met. Once trained, the random forest aggregates individual tree votes to make predictions, ensuring generalization and mitigating overfitting by leveraging diverse feature subsets. Thus, Random Forest Classification is a versatile and powerful tool for effectively handling complex classification tasks.

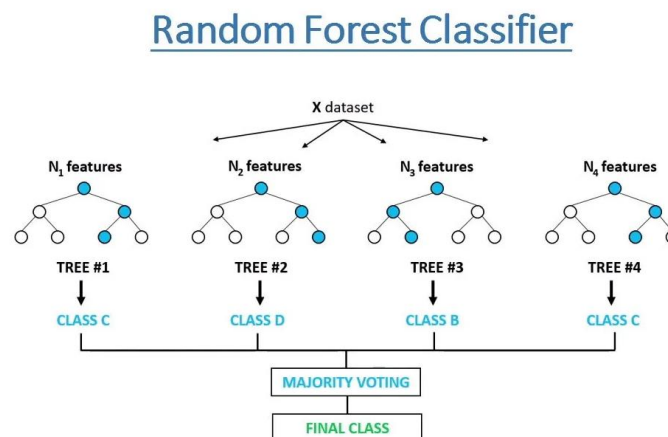


Figure 2.4 Random Forest Classifier

2.6 Summarization

One of the most difficult and challenging areas in Natural Language Processing (NLP) is automatic text summarization. It's a method for extracting a short and meaningful summary of text

from a wide range of sources, including books, news stories, blog posts, research papers, emails, and tweets. It can distill the most important information from a source to produce a shortened version for a particular user and task.

There are two main types of text summarization methods: extractive and abstractive. Extractive text summarization selects phrases and sentences from the source document to form the summary. Extractive summarization involves ranking the relevance of phrases to select the most relevant to the meaning of the source. Abstractive text summarization generates entirely new phrases and sentences to describe the contents of document. It can understand context and semantics and create own summary like human.

Existing NLP approaches for extractive text summarization such as scoring based term frequency, and cosine similarity have been popular in that time. Previous research approach tries to understand the significance of each sentence and their relationships with each other instead of understanding the context. Differently, abstract summarization approach tries to understand the content of the text. summary is provided based on context. In traditional NLP approaches, more complex linguistic models are involved because new sentences are created using template-based summarization.

2.6.1 Extractive Summarization

Extractive summarization is a method of summarizing a text document by selecting basic sentences or phrases from the original text and concatenating them to form a summary. It does not create new phrases or sentences but instead selects the most significant content from the original text. This method is known as “surface-level summarization” or “selection-based summarization”. Examples of extractive summarization include news articles, summaries of legal documents, and scientific papers. Extractive summarization can be used to quickly understand a large document’s main ideas and key points without having to read the entire text. This can save time and improve efficiency in many industries and applications, such as legal, medical, and business settings.

2.6.1.1 Cosine Similarity

In natural language processing, words can be represented as vectors to represent the meaning of words. Cosine is a popular method for calculating semantic similarity between two words, two sentences, or two documents, and it is a useful tool in practical applications (question answering, summarization, or automatic essay grading). Cosine similarity tests how close the two vectors A and B are based on their angle. Cosine similarity is the way to calculate semantic similarity between two words, two documents and it is important tool for natural language processing applications such as question answering, summarization.

The cosine similarity between the two vectors can be calculated using the formula:

$$\text{cosine similarity} = (\mathbf{v1} \cdot \mathbf{v2}) / (\|\mathbf{v1}\| * \|\mathbf{v2}\|)$$

where $\mathbf{v1}$ and $\mathbf{v2}$ are the vector representations of the sentences, and ‘ \cdot ’ denotes the dot product of two vectors. $\|\mathbf{v1}\|$ and $\|\mathbf{v2}\|$ are the Euclidean norms of the two vectors.

2.6.2 Abstractive Summarization

Abstractive summarization techniques create a short version of the original text by generating new sentences with words that are not necessarily found in the original text. Compared to extractive summarization, abstractive techniques are more daunting yet more attractive and flexible. Therefore, more and more attention is given to abstractive techniques in different languages. However, to the best of the knowledge, too few works have been dedicated to text summarization in the Myanmar language, of which almost all are extractive. This is partly due to the lack of proper Myanmar text datasets available for this task.

2.6.2.1 Sequence-to-Sequence Encoder-Decoder Model

In the realm of abstractive text summarization, various methodologies are employed, particularly in the context of the English language, with many relying on Sequence-to-Sequence (Seq2Seq) architectures. Text summarization is often conceptualized as a Seq2Seq task due to its nature. Seq2Seq models, a category of machine learning models utilized in natural language processing, offer a sophisticated approach to transforming input sequences into output sequences. These models are designed to grasp the intricate relationships between input data and desired

outputs, enabling them to produce coherent and novel text based on given inputs. Comprising two essential components, an encoder, and a decoder, Seq2Seq models operate by first processing the input sequence through the encoder to create a fixed-length vector representation, which is then utilized by the decoder to generate the corresponding output sequence. This framework forms the foundation for numerous advanced abstractive text summarization systems, showcasing their efficacy in understanding and generating meaningful summaries from textual data. Therefore, an approach to pre-train a Seq2Seq structure for text summarization can be quite promising.

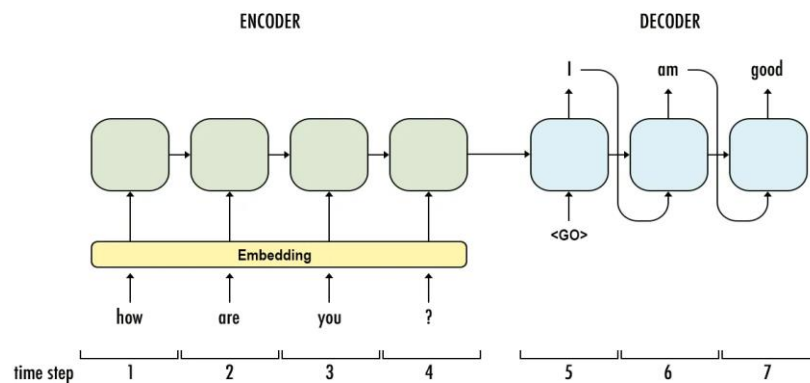


Figure 2.5 Example of seq2seq encoder-decoder architecture

Chapter 3 Methodology

3.1 Myanmar News Classification

The proposed text classification system for Myanmar news leverages supervised learning techniques to assign pre-defined category labels to text documents based on training data from Myanmar news websites. The methodology involves manual collection, labeling, and pre-processing of news articles in the training dataset. Feature selection is performed using the chi-square function, and a classification algorithm is implemented for text classification. The system utilizes Random Forest, Naïve Bayes, and SVM Classifier algorithms to conduct a comparative study on Myanmar data sets, aiming to identify the most effective approach for News Topic Classification.

3.1.1 System Architecture

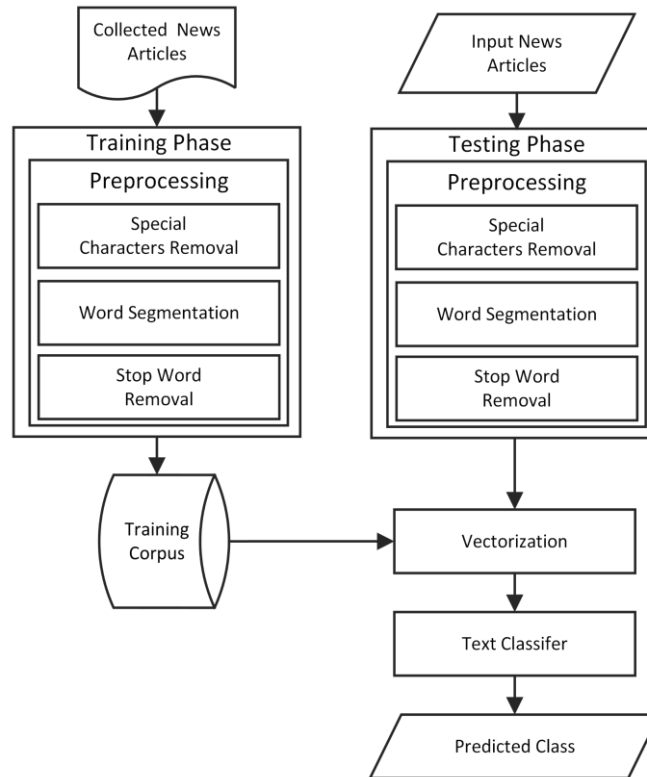


Figure 3.1 Proposed System Architecture for Myanmar News Classification

Figure 3.1 shows that the proposed system has two parts: training and testing. During the training phase, the collected raw news data undergoes preprocessing, and features are extracted. In the testing phase, feature words are extracted from the training corpus, followed by the classification process. The system defines four categories: politics, business, crime, and entertainment.

3.1.2 Data Collection

As the proposed system employs a supervised learning approach, it requires gathering raw data to create a training corpus. The dataset utilized for the Myanmar News Classification system is sourced from the Myanmar News dataset, collected from 7Day News and eleven Myanmar News websites. Initially, the original dataset comprised 5 categories, each containing 100 articles. However, the editorial category was excluded from the dataset, and 80 news articles were manually collected from each of the remaining categories across the eleven News website. As a result, the final dataset consists of 4 news categories: politics, business, crime, and entertainment, with each category comprising 180 news articles, totaling 720 articles.

Table 3.1 Description of Dataset Features

Features	Description	Type
Index	Index of News	(INT)
News	The News Article text	(STR)
Category	News Categories (politics, business, crime, and entertainment)	(STR)
category_id	Identification of News Categories	(INT) [0,3]

Index	News	Category	category_id
1	တီကျစ်ကျောက် မီးသွေးဓာတ်အားပေး စက်ရုံကို သက်တမ...	business	0
2	ရန်ကုန်တိုင်းအစိုးရ အစီအစဉ်ဖြင့် ရောင်းချမည့် ...	business	0
3	ရန်ကုန်မြို့သစ်စီမံကိန်း အတွက် လမ်းကြောင်းကွန်...	business	0
4	ပြည်တွင်း စက်သုံးဆီဈေး ရက်ပိုင်းအတွင်း ပြန်လည်...	business	0
5	ကချင်ပြည်နယ် မြစ် ကြီးနားနှင့် ချီဖွေမြို့အတွက်...	business	0

Figure 3.2 The Myanmar News Dataset

3.1.3 Data Preprocessing

Preprocessing is essential in text mining techniques and applications, serving as the initial step in the process. It significantly influences the quality of the classification stage by determining which keywords are meaningful and should be retained while eliminating those that do not contribute to distinguishing between the documents.

1. **Sentence Segmentation:** Sentence segmentation is the process of dividing the text into sentences, where each sentence forms a meaningful unit. In the formal Myanmar language, sentences are grammatically structured and typically end with the "။" pote-ma symbol.

Thus, when working with new articles, the "။" symbol is used to do sentence segmentation.

2. **Removing noise from Myanmar News text using Regex:** Noise removal involves eliminating both Burmese and English numbers, symbols, and special characters (example- ၂၀၂၃, ။, (,), hyperlinks, etc.) from the Myanmar News dataset using regular expressions.
3. **Lower-casing:** Lower-casing converts all the English text which are embedded within the Myanmar sentences (for example- person names, company names, and movie names) to lowercase to ensure uniformity in the processing of the text.
4. **Word-Segmentation:** In languages where words are separated by spaces, like English, it is simple to break down sentences into individual words. However, languages such as Japanese, Chinese, Thai, Hindi, and Burmese do not have clear spaces between words.

Therefore, word segmentation is essentially needed as a text preprocessing step. For this task, the "Pyidaungsu" Python library, which employs a technique called conditional random field (CRF) prediction to tokenize sentences into words is utilized.

5. **Stop word removal:** Stop words are removed from the collected documents. Myanmar stop words are တွင်၊ မယ်၊ မည်၊ သည်၊ သည့်၊ များ etc. These words have very low discrimination value since they occur in every Myanmar document. Hence, they do not help in distinguishing between documents with contents that are about different topics.

3.1.4 Feature Extraction and Selection

Feature extraction is the process of transforming raw data into a set of features that capture the essential characteristics of the data. This involves creating new features or representations from the original data in a way that retains as much relevant information as possible while reducing redundancy and noise.

The feature extraction process can be acquired by calling the *vectorizer.get_feature_names_out()*. The vectorizer is composed of using the **TFID** technique with **unigram and bigram as a hybrid** for feature extraction. Before testing with TFID vectorizer, apply other two types of vectorizations namely CountVectorizer and Word2Vec. When compared with TFID's performance with those two vectorizers, TFID performs the best. For this reason, TFID is chosen as the vectorizer for the system. This will generate all the features that the vectorizer got while fit transforming on the training data. Then, the occurrence count can be calculated on the features generated to view the most occurred features in the dataset. Since, the dataset contains four unique pairs of categories namely, **Business**, **Crime**, **Entertainment**, and **Politic**, features presented in these categories can be viewed separately.

1. Features of Business:

Category 1 Name: business
Category 1 Feature Counts:
Feature Name: သဘောထား, Occurrence Count: 56.7862275119906
Feature Name: ဝယ်, Occurrence Count: 40.80527373697426
Feature Name: အတွယ် ပြည့်, Occurrence Count: 37.09096892041893
Feature Name: ခရီးသွား, Occurrence Count: 33.358539386888395
Feature Name: လုပ်ငန်းရင်းနှီးမြှုပ်နှံ, Occurrence Count: 32.729465665984186
Feature Name: မောင်းပွဲ, Occurrence Count: 28.373783073521167
Feature Name: ဝယ်ယူ သုံးစွဲ, Occurrence Count: 27.74780338542768
Feature Name: ဧပြီ မတ်, Occurrence Count: 26.676074779753897
Feature Name: တရုတ်, Occurrence Count: 23.598804678715346
Feature Name: ငွေ စီမံ, Occurrence Count: 22.456529539130923

Figure 3.3 Vectorizer Features on Business Category

2. Features of Crime:

Category 2 Name: crime
Category 2 Feature Counts:
Feature Name: ရက်နေ့, Occurrence Count: 38.22554908707451
Feature Name: လမ်း, Occurrence Count: 28.507734409829123
Feature Name: ရဲ့ တပ်ပွဲ စစ်ဆေး, Occurrence Count: 22.386699440794224
Feature Name: အားထပ်မံ, Occurrence Count: 19.61885509793345
Feature Name: နှစ် နိုင်ငံ, Occurrence Count: 17.704237129378143
Feature Name: တိုင်, Occurrence Count: 16.698112284849024
Feature Name: မီးရှို့ ဖျက်ဆီး, Occurrence Count: 16.405702640957053
Feature Name: ဆေး ကုသ, Occurrence Count: 15.900679545183321
Feature Name: ဘွတ်ခံ, Occurrence Count: 15.66318407597886
Feature Name: အထက်တန်း, Occurrence Count: 15.283447554995073

Figure 3.4 Vectorizer Features on Crime Category

3. Features of Entertainment:

Category 3 Name: entertainment
Category 3 Feature Counts:
Feature Name: ရက်မှန်းလွဲ, Occurrence Count: 57.63445471901301
Feature Name: လတ်တလော, Occurrence Count: 52.321661906783866
Feature Name: တပ်, Occurrence Count: 33.33428014179668
Feature Name: ကြိုဆို, Occurrence Count: 28.151179033026246
Feature Name: ရက်နေ့, Occurrence Count: 23.244125931638425
Feature Name: ဇာတ်ကား စတင်, Occurrence Count: 19.18525290545908
Feature Name: ရက်နံနက်, Occurrence Count: 18.37159384571687
Feature Name: မဲ, Occurrence Count: 18.1721147945099
Feature Name: ဇာတ်ပို့ မင်းသား, Occurrence Count: 17.541790036259766
Feature Name: ရာခိုင်နှုန်းမြင့်, Occurrence Count: 15.703636942662744

Figure 3.5 Vectorizer Features on Crime Category

4. Features of Politic:

Category 4 Name: politic	
Category 4 Feature Counts:	
Feature Name: သစ်ပြည်သူ့	Occurrence Count: 55.185455612157845
Feature Name: အဖွဲ့	Occurrence Count: 46.70025638588513
Feature Name: အရိန်	Occurrence Count: 36.619051550701094
Feature Name: ပါတီ ရေးဆွဲ	Occurrence Count: 34.53602962885935
Feature Name: ဆွေးနွေးမလဲ	Occurrence Count: 31.115168757700484
Feature Name: ပြောင်းလဲ ဆေးဝါး	Occurrence Count: 28.686510635378887
Feature Name: အာဏာသိမ်း	Occurrence Count: 26.51362722884521
Feature Name: ကျယ်ပြန့်	Occurrence Count: 26.308518772786492
Feature Name: ကျော်စော သတင်း	Occurrence Count: 25.911059950922194
Feature Name: မစ္စတာ ဟုန်	Occurrence Count: 25.107606090555407

Figure 3.6 Vectorizer Features on Crime Category

Feature selection is the process of choosing a subset of the original features that are most relevant to the target variable or task at hand. The goal is to reduce the dimensionality of the data by removing irrelevant, redundant, or noisy features, thereby improving model performance, reducing overfitting, and speeding up computation.

The vectorized features have a total of **6980 features**. The number of features can be relatively reduced using the feature selection methods. The most decent and common approach is to apply ***Chi-Square Method***. Chi-square feature selection is a technique used in machine learning to select the most relevant features from a dataset. It works by measuring the dependence between each feature and the target variable using the chi-square test. The chi-square test measures the difference between the observed and expected frequencies of each feature and the target variable. The expected frequency is calculated based on the assumption that the feature and the target variable are independent. If the observed frequency is significantly different from the expected frequency, it indicates that the feature is dependent on the target variable and should be selected for the model. The chi-square statistic (χ^2) is calculated for each feature, representing the discrepancy between the observed and expected frequencies under the null hypothesis.

The formula for calculating the chi-square statistic for a contingency table is:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

Where:

O is the observed frequency.

E is the expected frequency under the null hypothesis.

To understand how many numbers of feature is best for the task, the feature selection process is evaluated with the help of the **SVM** model. Features ranging from 3000 to 6980 are fed into the SVM model and accuracies are calculated. According to the model output below, the feature count with a total of 4500 represent the best among others. But, for feature coverage of the system, a total of 5000 features is selected for the system.

```
k_best: 3000, with the shape (6769, 3000) has an Average Accuracy: 0.8255296780173389
k_best: 3500, with the shape (6769, 3500) has an Average Accuracy: 0.8296666633914895
k_best: 4000, with the shape (6769, 4000) has an Average Accuracy: 0.8308482381184762
k_best: 4500, with the shape (6769, 4500) has an Average Accuracy: 0.8327686928003966
k_best: 5000, with the shape (6769, 5000) has an Average Accuracy: 0.8320303587083139
k_best: 5500, with the shape (6769, 5500) has an Average Accuracy: 0.8296662267012088
k_best: 6000, with the shape (6769, 6000) has an Average Accuracy: 0.8289275650914156
k_best: 6570, with the shape (6769, 6570) has an Average Accuracy: 0.8271540566889488
```

Figure 3.7 Finding the best numbers of features using Chi-Square.

Both feature extraction and feature selection are critical steps in the machine learning pipeline, as they help improve model interpretability, generalization, and efficiency by reducing the complexity of the data while retaining its most informative aspects.

3.1.5 Myanmar News Topic Classification

To develop the Myanmar News Classification system, Machine Learning models can be utilized to give out the best Topic Classifiers. Among the ML models, Multinomial Naive Bayes (MNB), Linear Support Vector Machines (SVM), and Random Forest (RF) models can be utilized for the Topic Classification task.

3.1.5.1 Multinomial Naive Bayes (MNB)

After preprocessing the text data (cleaning, tokenization, etc.) and converting it into numerical feature vectors (e.g., TF-IDF vectors), Multinomial Naive Bayes is trained on a labeled dataset of news articles covering various topics (e.g., politics, sports, entertainment, business). The trained Multinomial Naive Bayes model learns the probability distribution of words/features for each category during the training phase. During the testing phase, the trained model computes the

probability of a news article belonging to each category based on its feature vector and selects the category with the highest probability as the predicted topic.

Multinomial Naive Bayes is computationally efficient and simple to implement. It performs well on text classification tasks, especially when the dataset is large and high-dimensional. Despite its "naive" assumption, Multinomial Naive Bayes often works surprisingly well in practice for many real-world text classification problems. MNB has been used in research and practical applications to achieve high accuracy in classifying news articles into different categories.

3.1.5.2 Support Vector Machine (SVM)

In Myanmar news classification, SVM can be used to classify news articles into different categories or topics based on their textual content. After preprocessing the text data and converting it into numerical feature vectors, SVM is trained on a labeled dataset of news articles covering various topics (politics, sports, entertainment, business). The trained SVM model learns the patterns and relationships between the features and the corresponding topics during the training phase. During the testing phase, the trained model is used to predict the topic of new, unseen news articles based on their feature vectors.

SVM is effective for high-dimensional data, making it suitable for text classification tasks where each word or term can be considered a feature. It works well with sparse data, which is common in text data where most feature values are zero (e.g., TF-IDF vectors). SVM has a solid theoretical foundation and tends to generalize well to new, unseen data.

3.1.5.3 Random Forest (RF)

Random Forest can be used to classify news articles into four categories: political, business, entertainment, and sports in Myanmar News Classification. The methodology involves collecting a diverse dataset of news articles, preprocessing the text data, vectorizing the text data using techniques like TF-IDF vectorization, training the Random Forest model, and evaluating the model's performance using metrics like precision, recall, F1-score, and accuracy. The trained

Random Forest model builds multiple decision trees, each of which learns different patterns and relationships between the features and the corresponding topics. During the testing phase, the trained model aggregates the predictions of all individual trees to predict the topic of new, unseen news articles.

Random Forest has been used in several studies for Myanmar News Classification, and it has shown promising results in terms of accuracy and performance. Random Forest is robust and less prone to overfitting compared to individual decision trees. It is a powerful algorithm that can handle high-dimensional data and is widely used in various applications, including text classification.

3.1.6 Grid Searching for SVM

Support Vector Machine (SVM) is a powerful supervised learning algorithm widely used for classification tasks, including natural language processing (NLP). However, the performance of an SVM model heavily depends on the selection of its hyperparameters. Grid search is a systematic approach used to tune the hyperparameters of machine learning models, including Support Vector Machines (SVMs), by exhaustively searching through a predefined hyperparameter space.

Grid search automates the process of hyperparameter tuning, eliminating the need for manual experimentation with different hyperparameter values. It evaluates all combinations of hyperparameters specified in the parameter grid, saving time and effort. By searching over a predefined grid of hyperparameters, grid search aims to optimize the model's performance metrics, such as accuracy, precision, recall, or F1 score. It allows for fine-tuning the model to achieve the best possible results.

Before applying grid search, the dataset needs to be preprocessed. Then, specify the hyperparameters to be tuned. These typically include the regularization parameter (C), the kernel type, and kernel-specific parameters such as the degree for polynomial kernels and gamma for RBF kernels. Next step includes defining a grid of hyperparameters to search over. This involves specifying the range of values for each hyperparameter.

It is essential to choose a reasonable range of values that cover a wide enough spectrum to ensure the search is comprehensive but not excessively large, as this would increase computation time. Finally, perform k-fold cross-validation to evaluate the performance of each combination of hyperparameters. Typically, values of k between 5 and 10 are used, where each fold is used as a validation set while the rest are used for training.

Once the grid search is complete, identify the combination of hyperparameters that resulted in the highest performance based on the chosen evaluation metric. These hyperparameters constitute the optimal configuration for the SVM model.

3.2 Myanmar News Summarization

The two primary categories of summarizing techniques are extractive and abstractive summarization. While abstractive summarizing creates the complete summary from scratch using the input text, extractive summarization chooses a subset of sentences from the original text to serve as the summary.

3.2.1 Extractive Summarization

Extractive summarization selects essential sentences from a text document to create a summary without generating new content. This method, known as "surface-level summarization," is commonly used for news articles, legal documents, and scientific papers. It allows for quick comprehension of the main ideas without reading the entire text. The proposed extractive text summarization employs cosine similarity to calculate sentence similarity and the TextRank algorithm for sentence selection.

3.2.1.1 System Architecture

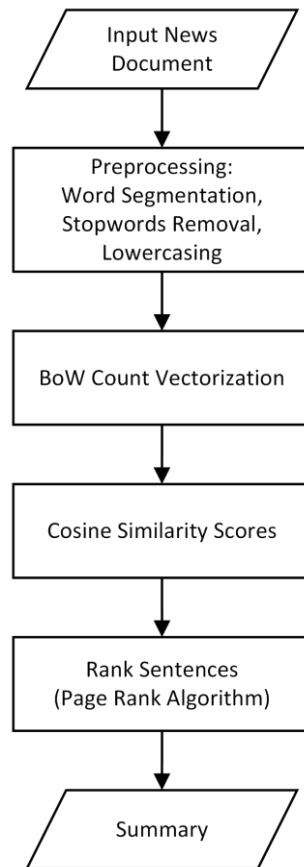


Figure 3. 8 Proposed System Architecture for Extractive Myanmar News Summarization

Figure 3.8 depicts the design of the proposed system. The proposed system has four main steps.

- Preprocessing step
- Vectorization
- Computing Cosine Similarity Scores
- Producing of summary sentence

Firstly, input news documents are segmented into words and stop words are removed in the preprocessing step. And then, each sentence in the article is vectorized utilizing the BoW count vectorization method. Cosine similarity scores between the sentences are computed and sentences are ranked by descending order according to their PageRank scores. Finally, the final summary is produced.

3.2.1.2 Data Preprocessing

The new article is split into sentences. Unlike English, Myanmar text does not use white space to separate words. It is also essential for all languages because it is the first step in linguistic processing. In this system, Myanmar news articles are segmented by the "Pyidaungsu" Python library which employs a conditional random field (CRF) prediction for splitting sentences to word. Stop words are unnecessary and unimportant information. Therefore, stop words are removed in preprocessing step. Moreover, all the English text which are embedded within the Myanmar sentences are converted to lowercase to ensure uniformity in the processing of the text.

3.2.1.3 Vectorization

The vectorization method used for the Extractive Myanmar News Summarization is “Bag-of-Words (BoW) Count Vectorization”.

Bag-of-Words (BoW):

BoW is a common technique in NLP where text data is represented as a "bag" (collection) of words, disregarding grammar, and word order. It focuses solely on the presence and frequency of words in a document or a sentence.

Count Vectorization:

Count vectorization is a specific method within BoW representation. It involves counting the occurrences of each word in a document or a sentence and representing the text as a vector where each component corresponds to the count of a particular word.

In this system, the BoW count vectorization was implemented by tokenizing sentences, constructing a vocabulary of unique words, and then representing each sentence as a vector of

word counts. This approach is a basic and commonly used method for text representation in NLP tasks such as text summarization, text classification, sentiment analysis, and information retrieval.

3.2.1.4 Cosine Similarity Scores

The cosine similarity considers the angle between the vectors of word frequencies for each document rather than just their magnitudes. This means that documents with similar word frequencies and distributions will have a smaller angle between their vectors and, thus a higher cosine similarity score. The goal is to quantify how closely related or similar each pair of sentences is in the context of the entire text. The cosine similarity between the two vectors can be calculated using the formula:

$$\text{cosine similarity} = (\mathbf{v1} \cdot \mathbf{v2}) / (||\mathbf{v1}|| * ||\mathbf{v2}||)$$

where $\mathbf{v1}$ and $\mathbf{v2}$ are the vector representations of the sentences, and ‘.’ denotes the dot product of two vectors. $||\mathbf{v1}||$ and $||\mathbf{v2}||$ are the Euclidean norms of the two vectors.

3.2.1.5 Sentence Selection

The similarity between each pair of sentences is calculated to create a similarity matrix. This matrix represents the similarity between each pair of sentences. Then, this similarity matrix is converted into a graph representation where each node represents a sentence, and the similarity scores represent the edge weights between nodes. The PageRank algorithm is applied on the graph to calculate a score for each node (sentence) based on the importance and connectivity of nodes in the graph. Nodes with higher scores are considered more important or central to the overall structure of the text. It iteratively updates the scores until they converge to stable values. The sentences are sorted in descending order according to their PageRank scores. Higher PageRank scores indicate sentences that are more central or important in the text. The top-ranked sentences are selected to form the summary. Including sentences with the highest PageRank scores ensures that the most important and informative sentences are included in the summary.

Table 3.2 Sample Input Article

Input article: 5 sentences	
News in Myanmar	<p>IS ပိုင် ရေနံတွင်းတွေကို ဗြိတိသျှ တိုက်လေယာဉ် ၄ စီးက တိုက်ခိုက်ခဲ့ ဗုံးကြဲ တိုက်ခိုက်ဖို့ အတွက် ယူကေ အမတ်တွေ အတည်ပြုပြီး မကြာမီမှာပဲ ဆိုက်ပရပ်စ်မှာ ရှိတဲ့ Akrotiri အက်ရောတီရီ လေတပ်စခန်းက တိုန်းဒိုး တိုက်လေယာဉ် ၄ စီး တိုက်ခိုက်မှုမှာ ပါဝင် ခဲ့ပါတယ်။ ဆီးရီးယား အရှေ့ပိုင်းက IS တွေ ထိန်းချုပ်ထားတဲ့ ရေနံတွင်းတွေကို တိုက်ခိုက် ထိမှန်ခဲ့တယ်လို့ ကာကွယ်ရေး ဝန်ကြီးက ပြောပါတယ်။ ဝန်ကြီးချုပ် ဒေးဗစ် ကင်မရွန်း ကတော့ IS အပေါ် တိုက်ခိုက်မှုတွေဟာ အချိန် ကြာမြင့်မှာ ဖြစ်ပြီး ပုံမှန် ပြုလုပ်သွားဖို့ လိုတယ်လို့ သတိပေး ပြောဆို ခဲ့ပါတယ်။ IS အပေါ် လေကြောင်း တိုက်ဖို့အတွက် ယူကေ ပါလီမန်မှာ ဗုဒ္ဓဟူးနေ့က ဆွေးနွေးငြင်းခုံပြီး မဲခွဲ ဆုံးဖြတ်ရာမှာ ထောက်ခံမဲ ၃၉၇ မဲ၊ ကန့်ကွက်မဲ ၂၂၃ မဲနဲ့ အတည်ပြုခဲ့တာ ဖြစ်ပါတယ်။ အက်ရောတီရီ လေတပ်စခန်းမှာ ရှိနှင့်ပြီးဖြစ်တဲ့ တိုက်လေယာဉ် ၈ စီး အပြင် နောက်ထပ် ၈ စီး ထပ်ပို့လိုက်တယ် လို့လည်း ယူကေ ကာကွယ်ရေး ဝန်ကြီး မိုက်ကယ် ဖာလန်က ပြောပါတယ်။</p>

Table 3.3 Ranked Sentences with their Cosine Similarity Scores between Sentence and PageRank Algorithm

Sentence ID		PageRank Scores
0	IS ပိုင် ရေနံတွင်းတွေကို ဗြိတိသျှ တိုက်လေယာဉ် ၄ စီးက တိုက်ခိုက်ခဲ့ ဗုံးကြဲ တိုက်ခိုက်ဖို့ အတွက် ယူကေ အမတ်တွေ အတည်ပြုပြီး မကြာမီမှာပဲ ဆိုက်ပရပ်စ်မှာ ရှိတဲ့ Akrotiri အက်ရောတီရီ လေတပ်စခန်းက တိုန်းဒိုး တိုက်လေယာဉ် ၄ စီး တိုက်ခိုက်မှုမှာ ပါဝင် ခဲ့ပါတယ်။	0.4757
1	ဆီးရီးယား အရှေ့ပိုင်းက IS တွေ ထိန်းချုပ်ထားတဲ့ ရေနံတွင်းတွေကို တိုက်ခိုက် ထိမှန်ခဲ့တယ်လို့ ကာကွယ်ရေး ဝန်ကြီးက ပြောပါတယ်။	0.2081
4	အက်ရောတီရီ လေတပ်စခန်းမှာ ရှိနှင့်ပြီးဖြစ်တဲ့ တိုက်လေယာဉ် ၈ စီး အပြင် နောက်ထပ် ၈ စီး ထပ်ပို့လိုက်တယ် လို့လည်း ယူကေ ကာကွယ်ရေး ဝန်ကြီး မိုက်ကယ် ဖာလန်က ပြောပါတယ်။	0.1784
3	IS အပေါ်လေကြောင်း တိုက်ဖို့အတွက် ယူကေ ပါလီမန်မှာ ဗုဒ္ဓဟူးနေ့က ဆွေးနွေးငြင်းခုံပြီး မဲခွဲ ဆုံးဖြတ်ရာမှာ ထောက်ခံမဲ ၃၉၇ မဲ၊ ကန့်ကွက်မဲ ၂၂၃ မဲနဲ့ အတည်ပြုခဲ့တာ ဖြစ်ပါတယ်။	0.0752
2	ဝန်ကြီးချုပ် ဒေးဗစ် ကင်မရွန်း ကတော့ IS အပေါ် တိုက်ခိုက်မှုတွေဟာ အချိန် ကြာမြင့်မှာ ဖြစ်ပြီး ပုံမှန် ပြုလုပ်သွားဖို့ လိုတယ်လို့ သတိပေး ပြောဆို ခဲ့ပါတယ်။	0.0628

Summary is produced as ascending order. Number of Top Rank Sentences can be chosen by user like (3 Sentences or 2 Sentences). In the above, Input article has 5 sentences and the two top-ranked sentences among 5 sentences are selected to form the summary if user choose 2 Sentences.

3.2.2 Abstractive Summarization

In contrast to extractive methods, abstractive summarizing techniques reduce information by creating new sentences. The steps involved in this approach are pre-processing the XL-Sum dataset, utilizing the data to finetune the mT5 model, and using the finetuned mT5 model to generate abstractive summaries.

3.2.2.1 System Architecture

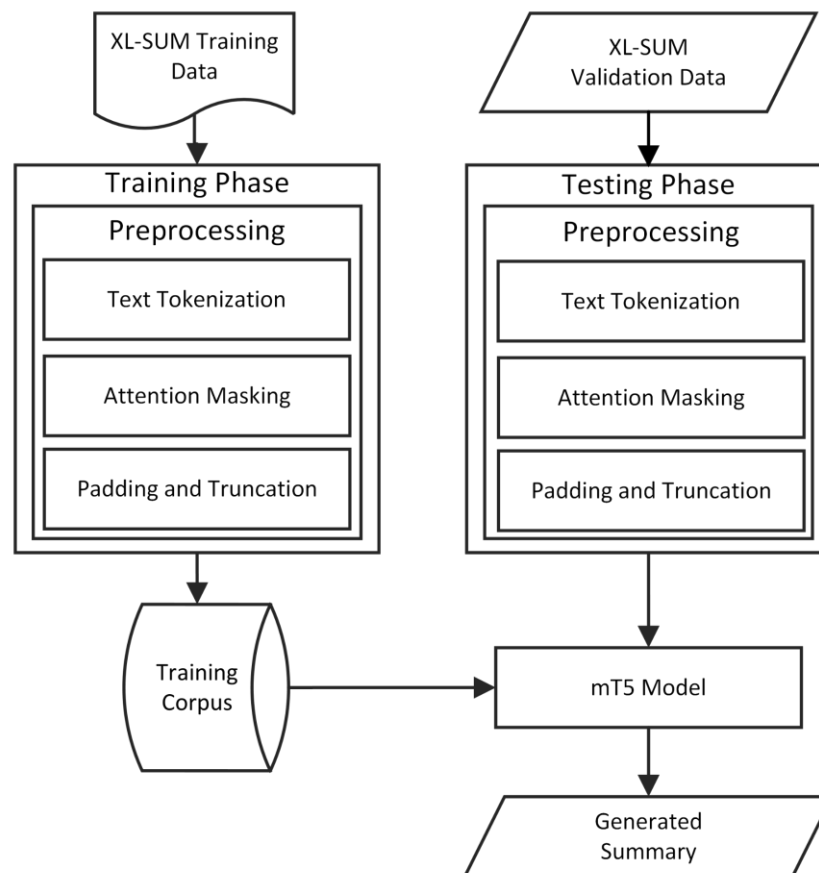


Figure 3.9 Proposed System Architecture for Abstractive Myanmar News Summarization

The architecture of the proposed abstractive text summarization system is shown in Fig. 2. Preprocessing the XL-Sum dataset, focusing solely on the Myanmar data is part of the process. The process includes preprocessing the dataset, optimizing the mT5 model on the prepared data, and generating summaries from the fine-tuned mT5 model.

3.2.2.2 Dataset

The dataset used for abstractive Summarization is the XL-Sum. It is a large-scale dataset and benchmark for extreme summarization, introduced by the researchers at Google Research. The news articles are crawled from the British Broadcasting Corporation (BBC) website. Using a custom crawler, the data has 1 million professionally annotated article-summary pairs covering 45 languages. Originating from a single source, these samples exhibit similar summarization strategies across all languages, making them ideal also for the multilingual summarization task. XL-Sum introduces the first publicly available summarization dataset and benchmarks for many languages (e.g., Bengali, Swahili). Thus, this dataset potentially enables and facilitates research on low resource languages, bringing technological advances to communities of these languages that have been traditionally under-served. Below table shows the languages covered by the XL-Sum dataset, and the number of samples for each language. Here, a sample denotes an article-summary pair.

Language	#Samples	Language	#Samples	Language	#Samples
Amharic	5,461	Korean	4,281	Somali	5,636
Arabic	40,327	Kyrgyz	2,315	Spanish	44,413
Azerbaijani	7,332	Marathi	11,164	Swahili	10,005
Bengali	8,226	Nepali	5,286	Tamil	17,846
Burmese	5,002	Oromo	5,738	Telugu	11,308
Chinese	39,810	Pashto	15,274	Thai	6,928
English	301,444	Persian	25,783	Tigrinya	4,827
French	9,100	Pidgin ^a	9,715	Turkish	29,510
Gujarati	9,665	Portuguese	23,521	Ukrainian	57,952
Hausa	6,313	Punjabi	8,678	Urdu	40,714
Hindi	51,715	Russian	52,712	Uzbek	4,944
Igbo	4,559	Scottish Gaelic	1,101	Vietnamese	23,468
Indonesian	44,170	Serbian (Cyrillic)	7,317	Welsh	11,596
Japanese	7,585	Serbian (Latin)	7,263	Yoruba	6,316
Kirundi	5,558	Sinhala	3,414	Total	1,005,292

Figure 3.10 Languages covered by the XL-Sum Dataset

The data is originally split into 80%-10%-10% (Train/Dev/Test) ratio. For Burmese, Train set includes 5761 articles while both Dev and Test set include 719 each. In total 7199 articles are composed to make the Burmese XL-Sum dataset. Data Fields include 'id' (A string representing

the article ID), '**URL**' (A string representing the article URL.), '**title**' (A string containing the article title.), '**summary**' (A string containing the article summary.), '**text**' (A string containing the article text.).

3.2.2.3 mT5 (multilingual T5)

mT5 was selected for fine-tuning for abstractive text summarization due to its unique characteristics from pre-trained language modeling methods that are commonly used. T5, denoting Text-to-Text Transfer Transformer, introduces a unified Seq2Seq framework designed to address diverse NLP text-based challenges. Notably, mT5, or Multilingual Text-to-Text Transfer Transformer, extends T5's capabilities by encompassing 101 different languages and leveraging a Common Crawl-based dataset for training. This multilingual aspect renders mT5 particularly advantageous for languages beyond English, including Myanmar.

The architecture of T5, as an encoder-decoder Transformer model, reflects the foundational structure of the Original Transformer model while addressing various objectives such as language modeling, de-shuffling, and predicting masked words within text spans. T5's innovative design enables the aggregation of multiple NLP tasks into a singular network, facilitated by its text-to-text format. Consequently, the same hyperparameters and loss function can be uniformly applied across different tasks.

mT5 inherits all the capabilities of T5 and was trained on an expanded version of the C4 dataset, encompassing over 10,000 web page contents across 101 languages, including Myanmar. When compared to alternative multilingual models like multilingual BERT and XLM-R, mT5 demonstrates superior performance across various tasks, particularly excelling in the summarization task.

3.2.2.4 Data Preprocessing

Data is tokenized by using the 250k word piece vocabulary provided with the mT5 checkpoint. The methodology behind the mT5 tokenizer involves a comprehensive process to prepare text data for the mT5 model, particularly when applied to languages like Myanmar. Here is a detailed expansion of the key steps involved:

1. **Vocabulary Construction:** The tokenizer constructs a vocabulary from the training corpus, encompassing a wide range of tokens including words, sub words, and special tokens. This vocabulary is crucial for encoding and decoding text inputs during model training and inference.
2. **Sub word Segmentation:** Byte Pair Encoding (BPE) is utilized for sub word segmentation, breaking down words into smaller sub word units. This technique helps the tokenizer handle rare or out-of-vocabulary words effectively by representing them as combinations of sub word tokens.
3. **Special Tokenization Rules:** Special tokenization rules are integrated into the tokenizer to address language-specific features and task requirements. For languages like Myanmar, these rules may include handling unique characters, diacritics, or linguistic structures that are specific to the language.
4. **Tokenization Process:** The tokenizer processes input text by segmenting it into tokens based on the constructed vocabulary and rules. It considers factors like word boundaries, punctuation marks, and whitespace to generate a tokenized representation of the text.
5. **Customization Options:** The tokenizer offers customization options such as setting maximum sequence length and padding tokens to ensure uniform input sizes for the model. These configurations help optimize memory usage and processing efficiency during training and inference.
6. **Compatibility with mT5 Model:** By following this methodology, the tokenizer ensures compatibility with the mT5 model architecture, allowing seamless integration for tasks like text summarization in languages like Myanmar. The tokenized input data is then fed into the model for further processing and generation of summaries.

Chapter 4 Implementation

4.1 Tools and Technologies used

Python has been selected as the programming language for the Myanmar News Classification and Summarization system. Several libraries have been utilized in this approach due to their extensive tools for machine learning. All experiments in this project will be carried out in Google Collab, a collaborative coding platform enabling Python code execution through the browser. For the development and deployment of the Website Application, PyCharm and Streamlit will be used, enabling the creation of an interactive and user-friendly interface. The python packages and versions used for development of the system can be seen in Table 4.1. Information about the hardware specifications for google colab can be found in Table 4.2.

Table 4.1 Python Libraries Version

Python Libraries Version	
Scikit-Learn	1.2.2
NumPy	1.25.2
Pandas	1.5.3
Matplotlib	3.7.1
Pydaungsu	0.1.4
Transformers	4.37.2
HuggingFace-Hub	0.20.3
Evaluate	0.4.1
Datasets	2.17.1
Torch	2.1.0+cu121
NLTK	3.8.1
Streamlit	1.31.1

Table 4.2 Google Collab Specifications

Google Collab Hardware Specs: CPU-only VMs	
Specification	Description
CPU	2.20 GHz
No. CPU Core	1
RAM	13 GB
Disk Space	108 GB

4.2 Myanmar News Classification

The implementation of the proposed text classification system for Myanmar news involves several key steps. Firstly, the system collects news articles from Myanmar news websites and manually labels them. These articles undergo pre-processing to clean and prepare them for analysis. Feature selection is performed using the chi-square function to identify relevant features. Finally, the system employs Random Forest, Naïve Bayes, and SVM algorithms for classification, aiming to compare their effectiveness for news topic classification.

4.2.1 Data

The experiment is conducted using the data collected from various Myanmar news websites, which includes both textual news data and speech transcriptions for all pre-defined categories. The dataset consists of 4 news categories: politics, business, crime, and entertainment, with each category comprising 180 news articles, totaling 720 articles. Each document within the dataset averages around six sentences. The concern is that using articles as input may lead to mixed topics within each article, making it challenging for the model to accurately classify them into a single category. Utilizing sentence-level data transformation from a whole article helps address class imbalances and improves error localization, enhancing overall classification accuracy. For this reason, all the articles are splitted into sentences and then sentences from each categories are evenly split into an 80-20 train-test ratio. The training set consists of 6,769 news sentences, while the test set comprises 1,695 news sentences, utilizing the 'scikit-learn' Python library for experimentation.

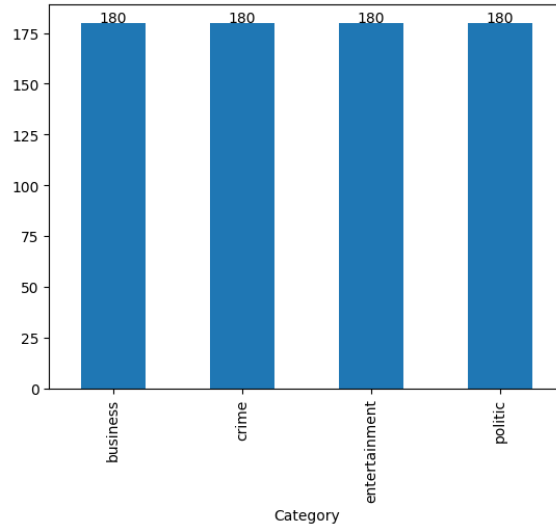


Figure 4.1 Myanmar News Articles

Table 4.3 Myanmar News Sentences

Data (Sentences)	Business	Crime	Entertainment	Politics
Training	1,492	1,429	2,056	1,752
Testing	374	358	514	439
Total	1,866	1,787	2,570	2,191

4.2.2 Data Preprocessing

Data Preprocessing is crucial within the text mining process, given its significant role. The dataset often contains various forms of redundant and unclean values, that are neither useful nor easy to model. During the preprocessing stage, tasks such as sentence segmentation, removal of numeric and special characters, lowercasing, word segmentation, and stop word removal are carried out.

4.2.2.1 Sentence Segmentation

In the formal Myanmar language, sentences are grammatically structured and commonly concluded with the "။" pote-ma symbol. It is assumed that the News Article data from the dataset follows this formal structure. Consequently, when working with new articles, the "။" symbol is employed for parsing sentences.

```
final_dict_list=[]
for obj in list_of_dicts:
    text = obj['News']
    sentences = text.split("။")
    # Remove any empty strings resulting from consecutive delimiters
    sentences = [sentence.strip() for sentence in sentences if sentence.strip()]
    my_dict_list = [{"Index":obj['Index'], "News": item, "Category":obj['Category'], "category_id":obj['category_id']} for item in sentences]
    final_dict_list.extend(my_dict_list)
data_list = [{"Index": i+1, 'News': item['News'], 'Category': item['Category'], 'category_id': item['category_id']} for i, item in
              enumerate(final_dict_list)]
data = pd.DataFrame(data_list)
```

Figure 4.2 Sentence Segmentation

4.2.2.2 Text Cleaning

In this stage, the text is cleaned by taking essential steps. Regex is used to remove numbers and various special characters, encompassing both Burmese and English symbols, from the Myanmar News data. This regular expression is designed to identify and remove sequences of digits (`\d+`) or a broad range of special characters and symbols, including Myanmar punctuation, common punctuation, and whitespace characters from text data. Then, all English text embedded within the Myanmar sentences is converted to lowercase to prevent any issues and ensure consistency in text processing.

```
import re

CleanPattern = re.compile(r'\d+|[!@#%&'>
def clean_sentence(sentence):
    sent = CleanPattern.sub(" ",str(sentence))
    return sent.lower()
```

Figure 4.3 Text Cleaning

4.2.2.3 Word Segmentation

The "Pyidaungsu" Python library, an open-source and freely available NLP Toolkit for the Myanmar language, is employed to tokenize sentences into words. In the Pyidaungsu library, word-level tokenization is supported only for Burmese and is based on conditional random field (CRF) prediction. While the word segmentation provided by the Pyidaungsu Library is generally effective for many words, some may be slightly inaccurate.

```
# Changing sentence into one word structure.
def tokenize(line):
    line = clean_sentence(line)
    sentence = pds.tokenize(line, form="word")
    sentence = ' '.join([str(elem) for elem in sentence])
    sentence = stop_word(sentence)
    return sentence

data['News'] = data['News'].apply(tokenize)
```

Figure 4.4 Word Segmentation

4.2.2.4 Stop word Removal

The proposed system receives input words that have been previously segmented and undergoes the removal of stop words. Among these stop words, there are approximately 390 collected words, including တွင်၊ မယ်၊ မည်၊ သည်၊ သည့်၊ များ။

```
#Load stopword file
stopwordlist = []
slist = []
with open("stopwords.txt", encoding = 'utf8') as stopwordsfile:
    stopwords = stopwordsfile.readlines()
    slist.extend(stopwords)
    for w in range(len(slist)):
        temp = slist[w]
        stopwordlist.append(temp.rstrip())

# Removed everything from the stopword list.
def stop_word(sentence):
    new_sentence = []
    for word in sentence.split():
        if word not in stopwordlist:
            new_sentence.append(word)
    return(' '.join(new_sentence))
```

Figure 4.5 Stopword Removal

4.2.3 Splitting Dataset

Every category of the Myanmar News data is equally divided into an 80-20 ratio for training and testing. This approach allows the model to be fed with an equal number of samples from each category, ensuring balanced performance across all categories.

```
train_data_list = []
test_data_list = []

# Iterate over each unique class and split the data
for category in labels.unique():
    # Filter data for the current category
    category_data = data[data['Category'] == category]
    # Split the data into train and test sets (80% train, 20% test)
    train_set, test_set = train_test_split(category_data, test_size=0.2, random_state=42)
    # Append the train and test sets to the respective lists
    train_data_list.append(train_set)
    test_data_list.append(test_set)

# Create DataFrames from the lists
train_data = pd.concat(train_data_list, ignore_index=True)
test_data = pd.concat(test_data_list, ignore_index=True)

# For Training
train_data = train_data[["News", "Category"]]

X_train = np.array(train_data["News"])
y_train = np.array(train_data["Category"])

# For Testing
test_data = test_data[["News", "Category"]]

X_test = np.array(test_data["News"])
y_test = np.array(test_data["Category"])
```

Figure 4.6 Splitting Dataset

4.2.4 Application of Vectorization

Before applying vectorization, define a function named `tokenize`. What this function does is it uses the `pds.tokenize` function from an external library (**pyidaungsu**) to split the line of text into separate words. The `form="word"` argument ensures that the function returns single words as tokens.

After initializing the `tokenize` function, pass that function as the `tokenizer` argument in the `TfidfVectorizer` object (**vectorizer**), this allows to use of specified tokenization logic. `TfidfVectorizer` is a class provided by **scikit-learn** that converts a collection of raw documents into a matrix of TF-IDF features. The `ngram_range=(1,2)` parameter specifies that both **unigrams** (single words) and **bigrams** (pairs of consecutive words) should be considered as features. The

min_df=3 parameter specifies that only words or bigrams occurring in **at least 3** documents will be considered as features. This helps filter out rare or noisy terms.

The `vectorizer.fit_transform(X_train)` fits the `TfidfVectorizer` on the training data (**X_train**) and transforms it into a TF-IDF matrix representation. The training data is both fitted and transformed because the TF-IDF vectorizer needs to learn the vocabulary from the training data.

The `vectorizer.transform(X_test)` transforms the test data using the same vectorizer that was fitted on the training data. It's crucial to use the same vectorizer (with the same vocabulary learned from the training data) for consistency between the training and test datasets. This ensures that the same features (words or bigrams) are used for both training and testing.

```
def tokenize(line):
    sentence = pds.tokenize(line,form="word")
    return sentence

vectorizer = TfidfVectorizer(tokenizer=tokenize, ngram_range=(1,2), min_df = 3)

X_train = vectorizer.fit_transform(X_train) # Fit and transform on the training data
X_test = vectorizer.transform(X_test) # Transform using the same vectorizer on the test data
```

Figure 4.7 Application of TF-IDF Vectorizer

4.2.5 Application of Chi-Square

Application of Chi-Square helps in selecting the most useful features from the vectorized features. A total of **6980 features** are generated by the vectorizer. To perform feature selection using the *chi-square* (χ^2) statistical test from the vectorized features, first different numbers of desired features (*k_best*) should be initialized. In this case, *k_best* is initialized with different numbers (desired features) such as 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, and 6980. Specified up to 6980 (total numbers of features from applying vectorizer). Next, various *k_best* values will be fed into the **Linear Support Vector Machine (SVM) model** to evaluate the performance of the SVM classifier with each *k_best* value (features) using cross-validation.

The `selector = SelectKBest(chi2, k=k_best)` creates a *SelectKBest* object with the *chi2* scoring function and the specified number of *k_best* features. The `X_train_k = selector.fit_transform(X_train, y_train)` fits the selector on the training data (**X_train**) and

transform it to select only the *k_{best}* most important features. This reduces the dimensionality of the feature space.

Finally, `svm_model = svm.SVC(kernel='linear')` creates a linear SVM classifier. The `scores = cross_val_score(svm_model, X_train_k, y_train, cv=5)` performs 5-fold cross-validation on the training data using the SVM classifier. This splits the training data into 5 folds, trains the model on 4 folds, and evaluates it on the remaining fold, repeating this process for each fold. At the end, the `average_accuracy = scores.mean()` calculates the average accuracy across all folds.

```
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import SelectKBest, chi2
from sklearn import svm, metrics

for k_best in [ 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 6980 ]:
    selector = SelectKBest(chi2, k=k_best)
    X_train_k = selector.fit_transform(X_train, y_train)

    # Train a model and evaluate using cross-validation
    svm_model = svm.SVC(kernel='linear')
    scores = cross_val_score(svm_model, X_train_k, y_train, cv=5) # 5-fold cross-validation
    average_accuracy = scores.mean()

    print(f"k_best: {k_best}, with the shape {X_train_k.shape} has an Average Accuracy: {average_accuracy}")
```

Figure 4.8 Searching *k_{best}* using SVM model.

After applying *k_{best}* with different numbers, *k_{best}* = 5000 (5000 features) came out as the highest score when fed into the SVM model. So, a total of 5000 features are filtered from the vectorizer and preserved as the best features for the Topic Classification task with the help of the **Chi-Square** feature selection method.

4.2.6 Models Applied for Topic Classification

As for the Topic Classification, different kinds of **Machine Learning (ML)** models can be used. But, depending on the nature of the dataset and the complexity of the task, three suitable ML models are evaluated and selected. The ML models include **Multinomial Naive Bayes (MNB)**, **Linear Support Vector Machine (Linear SVM)**, and **Random Forest (RF)**.

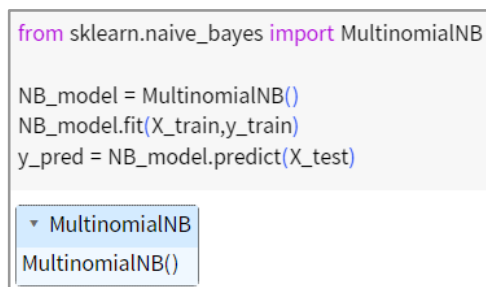
4.2.6.1 Multinomial Naive Bayes (MNB)

Multinomial Naive Bayes for text classification model, trains on the training data and then uses the trained model to predict the labels of the test data. This process allows for the classification of new, unseen text data into predefined categories or classes.

Firstly, import the Multinomial Naive Bayes classifier from *scikit-learn* by using **from sklearn.naive_bayes import MultinomialNB**. The **NB_model = MultinomialNB()** creates an instance of the Multinomial Naive Bayes classifier.

Then, initialize the **NB_model.fit(X_train, y_train)**, it trains the Multinomial Naive Bayes model on the training data. **X_train** represents the feature vectors of the training dataset, and **y_train** represents the corresponding labels or target values.

Finally, **y_pred = NB_model.predict(X_test)** uses the trained Multinomial Naive Bayes model to predict the labels of the test dataset (**X_test**). The predicted labels are stored in the **y_pred** variable to perform evaluation metrics later.



```
from sklearn.naive_bayes import MultinomialNB

NB_model = MultinomialNB()
NB_model.fit(X_train,y_train)
y_pred = NB_model.predict(X_test)
```

▼ MultinomialNB

MultinomialNB()

Figure 4.9 Application of Multinomial Naive Bayes (MNB)

4.2.6.2 Support Vector Machine (SVM)

Support Vector Machine for text classification model, trains on the training data and then uses the trained model to predict the labels of the test data. This process allows for the classification of new, unseen text data into predefined categories or classes.

Import the SVM classifier module from *scikit-learn* using **from sklearn import svm, metrics**. This enables to use of SVM for classification and later to evaluate the performance of the model. The **svm_model = svm.SVC(kernel='rbf')** creates an instance of the SVM classifier with

a rbf kernel. In SVM, the *kernel* specifies the type of decision boundary used to separate the classes. In this case, the rbf kernel decision boundary is a curve line in the feature space.

As for the model training, the `svm_model.fit(X_train, y_train)` trains the SVM classifier on the training data. The **X_train** represents the feature vectors of the training dataset, and **y_train** represents the corresponding labels or target values.

At last, model prediction can be done using the `y_pred = svm_model.predict(X_test)`. It uses the trained SVM model to predict the labels of the test dataset (**X_test**). The predicted labels are stored in the **y_pred** variable to perform evaluation metrics later on.

```
# Import classifiers and performance metrics
from sklearn import svm, metrics

# linear kernel model
svm_model = svm.SVC(kernel='rbf') # 'C': 1, 'gamma': 'scale' defaults
svm_model.fit(X_train, y_train)

# predict
y_pred = svm_model.predict(X_test)
```

Figure 4.10 Application of RBF Support Vector Machine (SVM)

4.2.6.3 Random Forest (RF)

Random Forest (RF) for text classification model, trains on the training data, and then uses the trained model to predict the labels of the test data. This process allows for the classification of new, unseen text data into predefined categories or classes. Random Forest is an ensemble learning method that combines multiple decision trees to improve classification performance and is widely used for various classification tasks due to its effectiveness and versatility.

Import the Random Forest classifier from scikit-learn using **from sklearn.ensemble import RandomForestClassifier**. This enables to use Random Forest for classification. The `rf_model = RandomForestClassifier(n_estimators=100, n_jobs=1)` creates an instance of the Random Forest classifier. The **n_estimators** parameter specifies the number of decision trees to be used in the forest. For topic classification, the classifier can use 100 decision trees. The **n_jobs**

parameter controls the number of jobs to run in parallel during both training and prediction. Here, it is set to 1, meaning it will do the job sequentially with only one CPU core.

As for the model training, the **rf_model.fit(X_train, y_train)** trains the Random Forest classifier on the training data. **X_train** represents the feature vectors of the training dataset, and **y_train** represents the corresponding labels or target values.

Finally, the model prediction can be done by using **y_pred = rf_model.predict(X_test)**. It uses the trained Random Forest model to predict the labels of the test dataset (**X_test**). The predicted labels are stored in the **y_pred** variable to perform evaluation metrics later.

```
from sklearn.ensemble import RandomForestClassifier

# Random forest classifier
rf_model = RandomForestClassifier(n_estimators=100, n_jobs=1)
rf_model.fit(X_train, y_train)

# predict
y_pred = rf_model.predict(X_test)
```

Figure 4.11 Application of Random Forest (RF)

4.2.7 Grid Searching for SVM

Grid search is a technique used for hyperparameter tuning in machine learning models. Hyperparameters are parameters that are not learned during the training process but are set before training. Grid search exhaustively searches through a specified set of hyperparameters, evaluating the model's performance using cross-validation on a training dataset for each combination. It then selects the combination of hyperparameters that yields the best performance.

First, import **GridSearchCV** from **sklearn.model_selection** and **SVC** (Support Vector Classifier) from **sklearn.svm**. Then, define **param_grid** which is a dictionary where keys are hyperparameters to be tuned ('C', 'kernel', 'gamma'), and values are lists of possible values for each hyperparameter. The 'C' is the regularization parameter in SVMs. It controls the trade-off between maximizing the margin and minimizing the classification error. A smaller value of 'C' leads to a softer margin, allowing some misclassifications in the training data. This can help prevent overfitting. A larger value of 'C' imposes a harder margin, penalizing misclassifications

more heavily. This may lead to better performance on the training data but could increase the risk of overfitting. Tuning 'C' allows balancing between bias and variance in the model.

The '**kernel**' specifies the type of kernel function used in the SVM algorithm. In this case, two kernels namely linear and rbf are defined. '**Gamma**' is a hyperparameter specific to certain kernel functions, particularly the RBF kernel. It defines the influence of a single training example, with low values meaning 'far' and high values meaning 'close'. A small value of 'gamma' implies a large similarity radius, leading to a smoother decision boundary. A large value of 'gamma' results in a tighter decision boundary, making the model more prone to overfitting.

GridSearchCV object (**grid_search**) is created. It takes **estimator** (the model to be tuned), **param_grid** (dictionary of hyperparameters), and **cv** (number of folds for cross-validation) as arguments. The fit method performs grid search with cross-validation of 5 and training the SVM model with each combination of hyperparameters. The **best_params** and **best_score** are obtained from the **grid_search** object. In this case, **Best Parameters** returns {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'} and **Best Score: 0.839711631573144**.

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

# Define the parameter grid to search
param_grid = {
    'C': [0.1, 1, 10],          # Regularization parameter
    'kernel': ['linear', 'rbf'], # Kernel type
    'gamma': ['scale', 'auto']  # Kernel coefficient for 'rbf' kernel
}

# Create SVM model
svm_model = SVC()

# Create GridSearchCV object
grid_search = GridSearchCV(estimator=svm_model, param_grid=param_grid, cv=5)

# Perform grid search on training data
grid_search.fit(X_train, y_train)

# Get the best parameters and best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best Score:", best_score)
```

Figure 4.12 Application of Grid Search for SVM

Based on the results from applying Grid Search algorithm on SVM, Radial Basis Function (RBF) Kernel will be chosen with the parameters of $C = 1$ and $\gamma = \text{scale}$. These hyperparameters will be used for model training of Myanmar News Classification Data and to make predictions.

4.3 Extractive Myanmar News Summarization

Extractive summarization is a method of summarizing text by selecting and extracting essential sentences directly from the original document without paraphrasing or generating new content. The implementation of extractive text summarization includes preprocessing the text, calculating sentence similarity using cosine similarity, applying the TextRank algorithm for sentence ranking, and selecting important sentences to form the summary.

4.3.1 Data Preprocessing

Before performing any analysis, the text data undergoes preprocessing stage. The preprocessing stage involves several key steps, including sentence tokenization, word segmentation, lowercasing and stop word removal. Sentence tokenization is achieved by splitting the text into individual sentences using predefined sentence separators. Myanmar news articles are segmented by the "Pyidaungsu" Python library for splitting sentences to word. Additionally, stop words, commonly occurring words with little semantic value are removed from each sentence to focus on content-bearing words. Finally, all the English text which are embedded within the Myanmar sentences are converted to lowercase.

```

SENTENCE_SEPARATOR = [".", "?", "!", ":", "...", "\n"]

def sentence_tokenize(text):
    sentences = re.split("(?<=[\"'.,:?!;'\"])\s*", text)
    if sentences[-1]:
        return sentences
    return sentences[:-1]

# Removed everything from the stopwords list.
def stop_word(sentence):
    new_sentence = []
    for word in sentence:
        if word not in stopwordslist:
            new_sentence.append(word)
    return(new_sentence)

def read_text(text):
    sentences = sentence_tokenize(text)
    return sentences

def sentence_similarity(sent1, sent2):

    sent1 = [w.lower() for w in pds.tokenize(sent1, form='word')]
    sent2 = [w.lower() for w in pds.tokenize(sent2, form='word')]
    sent1 = stop_word(sent1)
    sent2 = stop_word(sent2)

```

Figure 4.13 Data preprocessing for Extractive Summarization

4.3.2 Vectorization

After preprocessing, the text is transformed into a numerical representation suitable for analysis. The vectorization stage involves converting each sentence into a numerical vector using a bag-of-words approach. Firstly, all unique words across the dataset are identified to construct a vocabulary. Then, each sentence is represented as a vector, where each component corresponds to the frequency of a word in the vocabulary within that sentence.

```

all_words = list(set(sent1 + sent2))

vector1 = [0] * len(all_words)
vector2 = [0] * len(all_words)

for w in sent1:
    vector1[all_words.index(w)] += 1

for w in sent2:
    vector2[all_words.index(w)] += 1

```

Figure 4.14 Vectorization for Extractive Summarization

4.3.3 Cosine Similarity Scores

Once the text is vectorized, cosine similarity scores are computed to measure the similarity between pairs of sentences. Cosine similarity is a metric commonly used in text analysis to quantify the similarity between two vectors. For each pair of sentences, their vector representations are compared using cosine similarity, yielding a score between 0 and 1, where higher values indicate greater similarity.

```
if np.isnan(1 - cosine_distance(vector1, vector2)):
    return 0
return 1 - cosine_distance(vector1, vector2)

def build_similarity_matrix(sentences):

    similarity_matrix = np.zeros((len(sentences),len(sentences)))

    for idx1 in range(len(sentences)):
        for idx2 in range(len(sentences)):
            if idx1!=idx2:
                similarity_matrix[idx1][idx2] = sentence_similarity(sentences[idx1],sentences[idx2])
    return similarity_matrix
```

Figure 4.15 Computing Cosine Similarity Scores for Extractive Summarization

4.3.4 Sentence Selection

In the Sentence Selection stage, the top-ranked sentences are chosen based on their importance scores obtained from the PageRank algorithm. The goal is to select the most informative sentences for inclusion in the summary while adhering to the desired length constraint. These selected sentences are then assembled into the final summary. This stage ensures that the summary accurately captures the key points of the text.

```
import networkx as nx

def generate_summary(text,top_n):

    summarize_text = []

    sentences = read_text(text)
    sentence_similarity_matrix = build_similarity_matrix(sentences)
    sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_matrix)
    scores = nx.pagerank(sentence_similarity_graph)
    ranked_sentences = sorted(((scores[i],s) for i,s in enumerate(sentences)),reverse=True)
    if not ranked_sentences:
        return "" # Return empty summary and length 0
    else:
        for i in range(top_n):
            summarize_text.append(ranked_sentences[i][1].replace("\n", ""))
        return " ".join(summarize_text)
```

Figure 4.16 Sentence Selection for Extractive Summarization

4.4 Abstractive Myanmar News Summarization

Abstractive summarization involves creating a condensed version of a text by generating new sentences with words that may not appear in the original text. This method is more challenging but offers greater flexibility compared to extractive summarization techniques. In implementing abstractive summarization for Myanmar language, the MT-5 model is fine-tuned using XL-Sum Myanmar dataset, comprising articles and reference summaries.

4.4.1 Data Implementation

The system uses XL-Sum Burmese dataset in Hugging Face, which is the annotated article-summary pairs generated by BBC. This dataset has around 7000 samples for training, testing and validation.

```
from datasets import load_dataset  
  
ds = load_dataset("csebuethlp/xlsum", name="burmese")  
ds
```

Figure 4.17 XL-Sum data installation for Abstractive Summarization

After successful installation, the dataset will be available for further processing. In the downloaded dataset, there will be 4569 Training data, 570 Testing data and 570 Validation data. This data can be visualized by checking the first index. For example, the first data of training data can be seen by using the `ds["train"][0]`.

4.4.2 Implementation of Tokenization

```
from transformers import AutoTokenizer  
  
t5_tokenizer = AutoTokenizer.from_pretrained("google/mt5-small")
```

Figure 4.18 loading tokenizer in pre-trained google/mt5-small model.

This code initializes and utilizes the mT5 tokenizer from the Hugging Face Transformers library to tokenize input text and corresponding summaries. The tokenizer for the mT5 model is

loaded by calling the `AutoTokenizer.from_pretrained()` method with the model name `"google/mt5-small"`.

```
def tokenize_sample_data(data):
    input_feature = t5_tokenizer(data["text"], truncation=True, max_length=1024)
    label = t5_tokenizer(data["summary"], truncation=True, max_length=128)
    return {
        "input_ids": input_feature["input_ids"],
        "attention_mask": input_feature["attention_mask"],
        "labels": label["input_ids"],
    }

tokenized_ds = ds.map(
    tokenize_sample_data,
    remove_columns=["id", "url", "title", "summary", "text"],
    batched=True,
    batch_size=128)
tokenized_ds
```

Figure 4.19 Applying tokenization for dataset

The `tokenize_sample_data()` function is defined to tokenize the input data. It takes a dictionary data containing keys `"text"` and `"summary"`, representing the input text and its corresponding summary, respectively. Inside the `tokenize_sample_data()` function, the input text and summary are tokenized using the tokenizer's `__call__` method. The `truncation=True` argument specifies that the input should be truncated if it exceeds the maximum length allowed by the model. The `max_length` parameters set the maximum lengths for input text and summary. The function returns a dictionary containing the tokenized input text (`"input_ids"`) and its corresponding attention mask (`"attention_mask"`), as well as the tokenized summary (`"labels"`). The attention mask indicates which tokens should be attended to by the model during training.

The `map()` method is called on a dataset (`ds`) to apply the `tokenize_sample_data()` function to each sample in the dataset. The `remove_columns` argument specifies columns to be removed from the dataset after tokenization, as they are no longer needed. The `batched=True` and `batch_size` arguments enable processing the dataset in batches to improve efficiency. Finally, the tokenized dataset (`tokenized_ds`) is returned, containing the **tokenized input text**, **attention masks**, and **tokenized summaries** ready for training the mT5 model.

4.4.3 Optimization of Model Training


```

from transformers import AutoConfig, AutoModelForSeq2SeqLM

mt5_config = AutoConfig.from_pretrained(
    "google/mt5-small",
    max_length=128,
    length_penalty=0.6,
    no_repeat_ngram_size=2,
    num_beams=15,
)
model = (AutoModelForSeq2SeqLM
        .from_pretrained("google/mt5-small", config=mt5_config)
        .to(device))

```

Figure 4.20 Configuring the model

The code imports essential modules **AutoConfig**, **AutoModelForSeq2SeqLM**, and **DataCollatorForSeq2Seq** from the Transformers library. It sets up the configuration for the mT5 model by loading the configuration for the "google/mt5-small" model with specific parameters like `max_length`, `length_penalty`, `no_repeat_ngram_size`, and `num_beams`. The model is then moved to a specified device (GPU) for computation.

```

from transformers import DataCollatorForSeq2Seq

data_collator = DataCollatorForSeq2Seq(
    t5_tokenizer,
    model=model,
    return_tensors="pt")

```

Figure 4.21 Applying Data Collator

For the sequence-to-sequence (seq2seq) task, there is a need to stack the inputs for the encoder and prepare for the decoder side. In seq2seq setup, a common technique called "teach forcing" will then be applied in the decoder. These tasks are not needed to manually set up in Hugging Face, and **DataCollatorForSeq2Seq** handles all necessary steps. In this collator, the padded token will also be filled with label id -100. The data collator also utilizes the mT5 tokenizer (`t5_tokenizer`), associates it with the model, and specifies to return PyTorch tensors (`return_tensors="pt"`).

4.4.4 Implementation of ROUGE Score evaluation

```

import evaluate
import numpy as np

rouge_metric = evaluate.load("rouge")

def tokenize_sentence(arg):
    encoded_arg = t5_tokenizer(arg)
    return t5_tokenizer.convert_ids_to_tokens(encoded_arg.input_ids)

def metrics_func(eval_arg):
    preds, labels = eval_arg
    # Replace -100
    labels = np.where(labels != -100, labels, t5_tokenizer.pad_token_id)
    # Convert id tokens to text
    text_preds = t5_tokenizer.batch_decode(preds, skip_special_tokens=True)
    text_labels = t5_tokenizer.batch_decode(labels, skip_special_tokens=True)
    # Insert a line break (\n) in each sentence for ROUGE scoring
    text_preds = [(p if p.endswith(("!", "!", "?", "? ", ". ", " ", " ")) else p + ". ") for p in text_preds]
    text_labels = [(l if l.endswith(("!", "!", "?", "? ", ". ", " ")) else l + ". ") for l in text_labels]
    sent_tokenizer_my = RegexpTokenizer(u'^!! ?? . ||]*!! ?? . ||]')

    text_preds = ["\n".join(np.char.strip(sent_tokenizer_my.tokenize(p))) for p in text_preds]
    text_labels = ["\n".join(np.char.strip(sent_tokenizer_my.tokenize(l))) for l in text_labels]
    # compute ROUGE score
    return rouge_metric.compute(
        predictions=text_preds,
        references=text_labels,
        tokenizer=tokenize_sentence
    )

```

Figure 4.22 prepare metrics function for ROUGE Score evaluation

Metrics functions for evaluating training are prepared as well. Typically, metrics like BLEU and ROUGE are employed. However, for summarization tasks, **ROUGE** is preferred as it evaluates the inclusion of essential words from the reference text in the generated output. ROUGE extends BLEU by also considering n-grams from the reference text in the generated output, emphasizing recall. Variations like **ROUGE-L** and **ROUGE-Lsum** further measure the longest common substrings. In Hugging Face, built-in objects handle these metrics, eliminating the need for manual implementation. The provided code tokenized sentences with the mT5 tokenizer, processes text predictions and labels, and computes ROUGE scores. Additionally, it handles special cases such as replacing padded token IDs and ensuring proper formatting for ROUGE scoring.

4.4.5 Implementation of Data Loader

A **DataLoader** in PyTorch is an iterable that allows batching and parallelizing data loading for training and evaluation. It is commonly used in machine learning pipelines to efficiently load and process data during model training or evaluation.

```

from torch.utils.data import DataLoader

sample_dataloader = DataLoader(
    tokenized_ds["test"].with_format("torch"),
    collate_fn=data_collator,
    batch_size=5)
for batch in sample_dataloader:
    with torch.no_grad():
        preds = model.generate(
            batch["input_ids"].to(device),
            num_beams=15,
            num_return_sequences=1,
            no_repeat_ngram_size=1,
            remove_invalid_values=True,
            max_length=128,
        )
    labels = batch["labels"]
    break

metrics_func([preds, labels])

```

Figure 4.23 Implementation of DataLoader

Import the **DataLoader** class from the **torch.utils.data** module, which is part of the PyTorch library. The **sample_dataloader** is initialized as a **DataLoader** object. The dataset is loaded into batches and changed into torch format. The **collate_fn=data_collator** is used to collate and batch individual samples into mini batches. Then, the input to the generate method includes the input IDs (**batch["input_ids"]**) from the mini batch, which are transferred to the appropriate device (e.g., GPU) for computation. Various parameters are passed to the generate method to control the generation process, such as the number of beams, maximum length, and whether to remove repeated n-grams. Then, the labels for the mini batch (**batch["labels"]**) are extracted. Note that, in order to avoid suboptimal text generation, applied the beam search for the text generation. Finally, **metrics_func** is called with the generated predictions (preds) to evaluate the rough scores.

4.4.6 Finetuning using Seq2SeqTrainer Class

The **Seq2SeqTrainer** class provides a high-level interface for training seq2seq models, handling various aspects of the training process to streamline and simplify the workflow. By setting of **predict_with_generate=True** in **Seq2SeqTrainingArguments**, the predicted tokens generated by **model.generate()** will be used in each evaluation. In order to evaluate ROUGE score

using the predicted tokens the system used built-in Seq2SeqTrainer classes in HuggingFace, instead of usual TrainingArguments and Trainer.

```
from transformers import Seq2SeqTrainer
trainer = Seq2SeqTrainer(
    model = model,
    args = training_args,
    data_collator = data_collator,
    compute_metrics = metrics_func,
    train_dataset = tokenized_ds["train"],
    eval_dataset = tokenized_ds["validation"].select(range(20)),
    tokenizer = t5_tokenizer,
)
trainer.train()
```

Figure 4.24 Implementation of Seq2Seq Trainer Class

The **Seq2SeqTrainer** class is instantiated to manage the training process. It takes several arguments such as **training_args** from **Seq2SeqTrainingArguments**, **data_collator** which is responsible for batching and padding input sequences, **compute_metrics** for computing evaluation metrics (**ROUGE score**), and finally **training set** and **evaluation set**, which in this case evaluation is set to 20 for efficiency because of high computational cost of evaluation (ROUGE scoring). After the requirements are set, it is executed by calling the **train()** method of the trainer object (trainer) to start the training process.

During training, the model will be fine-tuned on the specified training dataset (train_dataset), and evaluation will be performed periodically based on the evaluation strategy. The training loop, including backpropagation, gradient updates, and evaluation, is handled internally by the **Seq2SeqTrainer**, simplifying the training process.

Chapter 5 Evaluation of Experimental Results

5.1 Myanmar News Classification

The performance of the test set is evaluated using precision, recall, and F-measure metrics. The system assigns each news sentence to a single category based on its content, allowing for the calculation of precision, recall, and F-measure for each category. Evaluation metrics are crucial for assessing the performance of Text Classification models, using parameters like True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) in their computation. Accuracy, precision, recall, and F1-score are determined as follows:

Accuracy: $(TP + TN) / (TP + TN + FP + FN)$

Precision: $TP / (TP + FP)$

Recall (Sensitivity or True Positive Rate): $TP / (TP + FN)$

F1-Score: $2 * (Precision * Recall) / (Precision + Recall)$

In the context of text classification, these metrics play important roles:

Accuracy:

Accuracy serves as a suitable evaluation measure for well-balanced classification problems without significant skewness or class imbalance.

Precision:

Precision assesses the ratio of the number of correctly classified news sentences to that category to the total number of news sentences labeled by the system as that category. It is particularly valuable when minimizing False Positives is a priority.

Recall:

Recall measures the ratio of correctly classified news sentences to the total number of news sentences of a specific category in the training set. High recall, ideally equal to 1, is desirable, especially in scenarios where avoiding False Negatives is critical.

F1-Score:

The F1 score, ranging from 0 to 1, represents the harmonic mean of precision and recall, providing a balanced measure of a model's performance.

5.1.1 Performance Evaluation of Naive Bayes Classifier

After performing the confusion matrix, the Naive Bayes model has a good accuracy of **84%**. The Precision, Recall, and F1 Score of the Naive Bayes Model are as follows.

Table 5.1 Experimental Results for Naive Bayes Classifier

	Business	Crime	Entertainment	Politics
Precision	0.88	0.83	0.81	0.84
Recall	0.82	0.80	0.90	0.80
F1 Score	0.85	0.82	0.85	0.82

5.1.2 Performance Evaluation of SVM Classifier

After analyzing the confusion matrix, the SVM model achieves a good accuracy level of **83%**. The Precision, Recall, and F1 Score of the SVM Model are as follows.

Table 5.2 Experimental Results for SVM Classifier

	Business	Crime	Entertainment	Politics
Precision	0.87	0.85	0.77	0.86
Recall	0.82	0.77	0.91	0.78

F1 Score	0.84	0.81	0.84	0.82
-----------------	------	------	------	------

5.1.3 Performance Evaluation of Random Forest Classifier

Following the confusion matrix analysis, the RF model has an accuracy of **78%**. The Precision, Recall, and F1 Score of the RF Model are as follows:

Table 5.3 Experimental Results for Random Forest Classifier

	Business	Crime	Entertainment	Politics
Precision	0.81	0.77	0.76	0.79
Recall	0.78	0.70	0.89	0.71
F1 Score	0.79	0.74	0.82	0.75

5.1.4 Comparison of the Accuracies with Various Vectorization Methods

Three different kinds of vectorization were applied for the Myanmar News Classification Task. Different Vectorization results in different performance accuracies. It was found that TFIDF is the most efficient vectorizer, demonstrating effective performance across all three supervised machine learning models according to the findings below.

Table 5.4 Compare Accuracies with Various Vectorizers

Accuracy	TFID	CountVec	W2Vec
SVM	0.83	0.78	0.60
Naïve Bayes	0.84	0.84	0.37
RF	0.78	0.78	0.66

Both TFID and Count Vectorizer techniques convert text data into numerical vectors and they both produce sparse representations of the text data. Even though they are like each other, depending on the nature of the dataset used and the desired task, TFID came out as the best option. As for the W2Vec Vectorizer, it has been successful in various NLP tasks for many languages, including English, but it may not scale well for languages with limited linguistic resources, such as Myanmar. Because Word2Vec typically handles out-of-vocabulary words poorly. In languages with limited linguistic resources, the likelihood of encountering out-of-vocabulary words is higher due to the limited vocabulary coverage. For these reasons, TFID is selected as the best vectorizer for the Myanmar News classification task.

5.1.5 Comparison of the Accuracies with Various ML Methods

To compute the performance of the Myanmar News Classifier, various Machine Learning Models such as Naive Bayes, Support Vector Machine, Random Forest, Decision Tree, and K-Nearest Neighbors classifiers are used.

Table 5.5 Accuracies of Different Models with Myanmar News Dataset

	Naïve Bayes	SVM	Random Forest	Decision Tree	KNN	Best Score
Accuracy	0.84	0.83	0.78	0.69	0.60	Naïve Bayes
Precision	0.84	0.83	0.78	0.70	0.67	Naïve Bayes
Recall	0.84	0.83	0.78	0.69	0.60	Naïve Bayes
F1 Score	0.84	0.83	0.78	0.69	0.61	Naïve Bayes

The naive Bayes classifier is found to be most efficient concerning the number of features and the vectorization method used. The SVM on the other hand, is observed to be the second top-performing classifier. But, even though MNB is computationally efficient and best performing, the independence assumption of MNB might not always hold in real-world text data.

MNB may struggle to capture complex relationships present in real-world data.

On the other hand, SVMs perform well in high-dimensional spaces, making them suitable for text classification with many features. SVMs can capture complex relationships in the data using kernel functions. SVMs can be robust against overfitting, especially in high-dimensional spaces. For these reasons, SVM is selected as the most suitable model for Myanmar News Classification.

Apart from SVM and MNB, the remaining models also perform well for the Myanmar News Classification task. However, their accuracies are lower than the SVM and MNB. This causes them to not choose the best-performing model for the classification task.

The performance in the classification process is affected by the amount of training corpus, word segmentation, and the stop words selected. These issues cause the proposed system's ML models to limit their overall performance. In the experiment, a combination of the SVM algorithm and TF-IDF method is chosen as the best practice for the Myanmar News dataset.

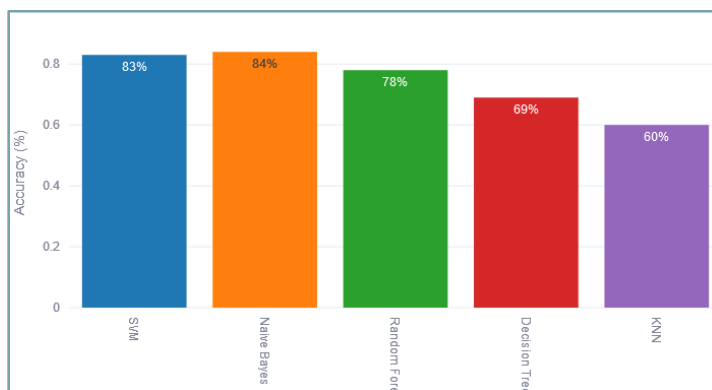


Figure 5.1 Comparison of the Accuracies with Various Models for Classification

5.2 Myanmar News Summarization

In summarization research, evaluation of the quality of a generated summary is important. The task of evaluating a summary is difficult because there is no perfect summary for a document, and the meaning of a good summary is an open question to a significant extent. Human summarizers have been found to have poor cooperation when it is evaluating and generating

summaries. Furthermore, the widespread use of different metrics, as well as the lack of a standard evaluation metric, has made summary evaluation difficult and challenging.

One of the metrics for evaluation of text summarization is recall-oriented understudy for gisting evaluation (ROUGE). It works by comparing human summary (one or several reference summaries) and system summary based on n-grams. ROUGE is a popular method for calculating the efficiency of auto-generated summaries. Precision in ROUGE means that how much of the system summary was relevant. Recall in ROUGE simply means how much of the reference summary is the system summary recovering or capturing. F1-SCORE in ROUGE means a balance between recall and precision, giving a single value to gauge overall performance. Precision, Recall, and F1-score in ROUGE are determined as follows:

$$\text{Recall} = \frac{\text{number_of_overlapping_words}}{\text{total_words_in_reference_summary}}$$

$$\text{Precision} = \frac{\text{number_of_overlapping_words}}{\text{total_words_in_system_summary}}$$

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}$$

ROUGE-1 refers to the overlap of unigrams between the system summary and the reference summary. ROUGE-2 refers to the overlap of bigrams between the system and reference summaries. The overlap of unigrams, bigrams, trigrams, and higher order n-grams is measured by ROUGE-N. It is recall-based measures and based on n-grams comparison.

ROUGE-L works the concept of longest common subsequence (LCS) between the two sequences of text. It has the advantage of not requiring consecutive matches, but in-sequence matches that represent sentence level word order. ROUGE-Lsum initially divides the summaries into sentences and then evaluates the ROUGE-L scores for each sentence separately. Subsequently, it aggregates these scores and calculates the average to determine the overall ROUGE-Lsum Score.

5.2.1 Performance Evaluation of Extractive Myanmar News Summarization

In evaluating performance, the Extractive Myanmar News Summarization is applied to a set of 3,000 Myanmar news articles sourced from the XL-Sum Dataset, which contains annotated reference summaries. The resulting ROUGE-1, ROUGE-2, and ROUGE-L scores of the system are as follows.

Table 5.6 Experimental Results for Extractive Summarization

	Precision	Recall	F1-Score
ROUGE-1	39.73	51.24	43.14
ROUGE-2	15.91	23.85	18.1
ROUGE-L	27.77	35.72	30.04

The extractive summarization system demonstrated a commendable performance with moderate Precision, Recall, and F-1 Score values across various ROUGE metrics. Notably, the system achieved significant scores for ROUGE-1, ROUGE-2, and ROUGE-L, indicating its effectiveness in capturing the essence of the original text. To further elevate its capabilities, integrating contextual embeddings or advanced deep learning models could enhance the system's ability to capture the nuanced context of the text.

5.2.1 Performance Comparison of Multilingual Model and the proposed Monolingual Model of Abstractive Summarization on XL-Sum Dataset

The comparison for Abstractive Summarization involves the implemented proposed Monolingual Model, which is fine-tuned solely on the Myanmar dataset from XL-Sum Dataset, and a Multilingual Model fine-tuned with all the 45 languages of XL-Sum Dataset as the baseline. The resulting ROUGE-1, ROUGE-2, and ROUGE-L scores of the systems are as follows.

Table 5.7 Accuracies of multilingual model and monolingual model with XL-Sum Dataset

	Multilingual Model	Monolingual Model
ROUGE-1	15.96	36.67
ROUGE-2	5.15	16.78
ROUGE-L	14.18	29.98

In comparing the performance of the multilingual and monolingual models based on the ROUGE metrics, it is evident that the monolingual model outperforms the multilingual model. Specifically, the monolingual model achieves higher ROUGE scores, indicating its superior ability to capture the essence of the original text and generate more accurate and comprehensive summaries. This underscores the importance of language-specific fine-tuning for achieving optimal performance in text summarization tasks.

5.2.2 Performance Comparison of Low-Resource mT5 Model and the proposed Monolingual Model of Abstractive Summarization on XL-Sum Dataset

The assessment of Abstractive Summarization comprises the proposed Monolingual Model, specifically trained on the Myanmar dataset from XL-Sum Dataset, and a low-resource model employing mT5 without fine-tuning. The resulting ROUGE-1, ROUGE-2, and ROUGE-L scores of the systems are as follows.

Table 5.8 Accuracies of low-resource model and monolingual model with XL-Sum Dataset

	Low-Resource Model	Monolingual Model
ROUGE-1	20.26	36.67
ROUGE-2	10.46	16.78
ROUGE-L	19.30	29.98

Compared to the low-resource model, the monolingual model performed better since it was trained exclusively on the Myanmar dataset, which improved its comprehension and summarization of the language.

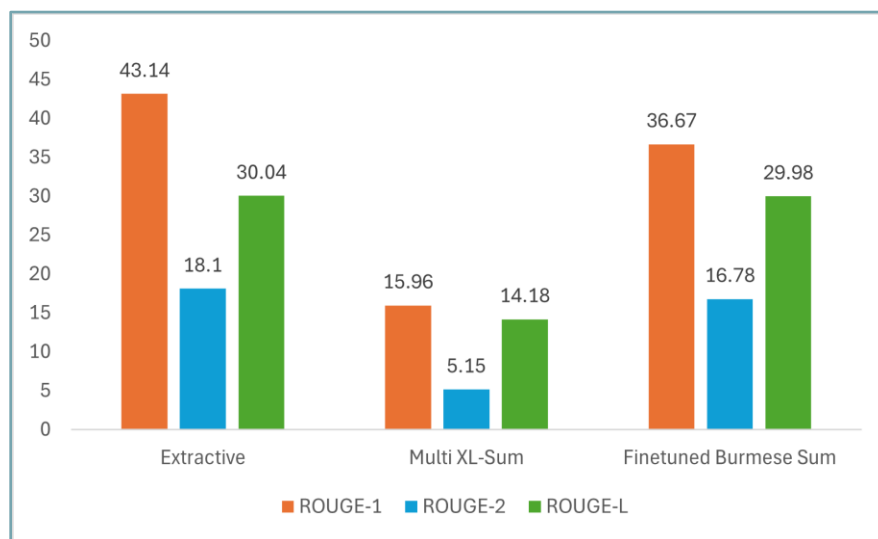


Figure 5.2 Comparison of the Accuracies with Various Models for Summarization

Chapter 6 Application Implementation

For the proposed system, Streamlit was used to build the website, while Streamlit Cloud was employed for the website development. Streamlit, a framework for interactive web applications, enabled easy website creation. Streamlit Cloud, a platform for deploying and sharing Streamlit apps, efficiently supported development. This approach ensured smooth progress from website creation to development, enhancing workflow efficiency and effectiveness.

6.1 Streamlit Cloud

Deploying website applications can be challenging, particularly in terms of handling server infrastructure and scaling to meet user demands. Streamlit Cloud addresses these challenges by providing a platform that streamlines the deployment and sharing of Streamlit apps. It eliminates the need to set up servers, manage dependencies, or deal with deployment complexities. The process of deploying a Streamlit app on Streamlit Cloud follows the following diagram.

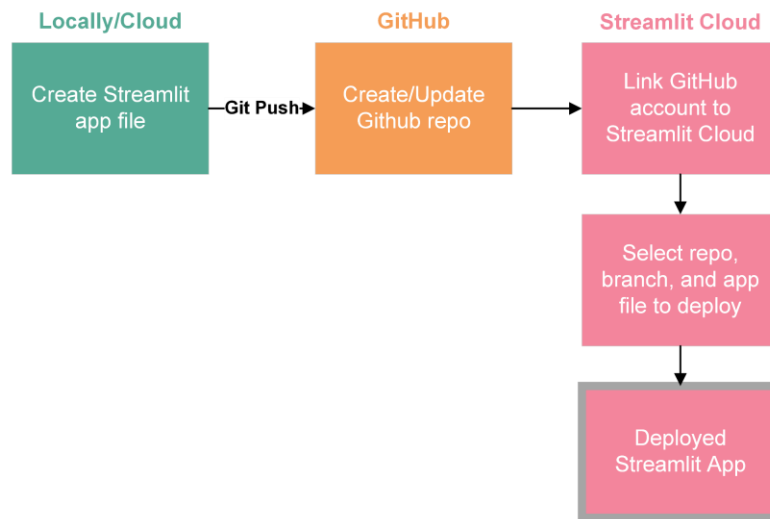


Figure 6.1 Streamlit Cloud Process Flow

With the help of Streamlit Cloud, users can access the deployed system on multiple platforms via the Internet. All the user needs are to have the URL link of the system and once they have it, they can make multiple News Article predictions and summarizations in real-time.

6.2 Myanmar News Classification

In the Myanmar News Classification Part, the SVM and NB models were utilized to make predictions from the users. When the user enters the website, the user can access the Home page. By going to the Navigation section, the user will be able to go to the "Classification" page. First, the user can make predictions in the Text Box given. After typing the News article in the given textbox, kindly click on the Predict" button. Once the processing is done, the user will be able to see the category of the user's input via a message.

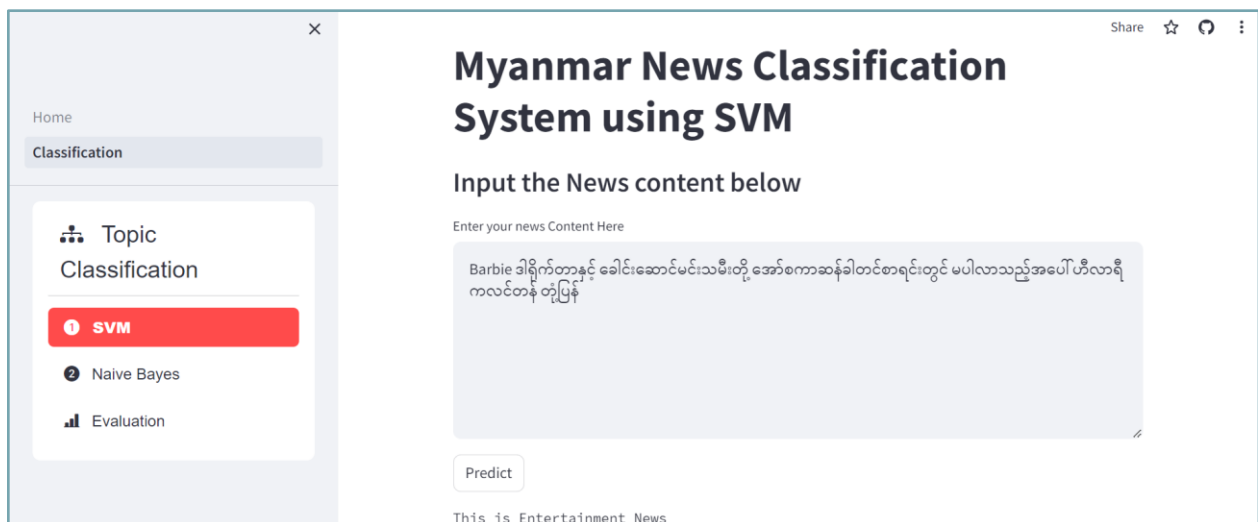


Figure 6.2 Myanmar News SVM Classifier's Interface

Moreover, the user can also choose the type of models in the navigation section. By default, SVM will be used for prediction. However, users can apply the Naive Bayes model by clicking on the "Naive Bayes" in the navigation.

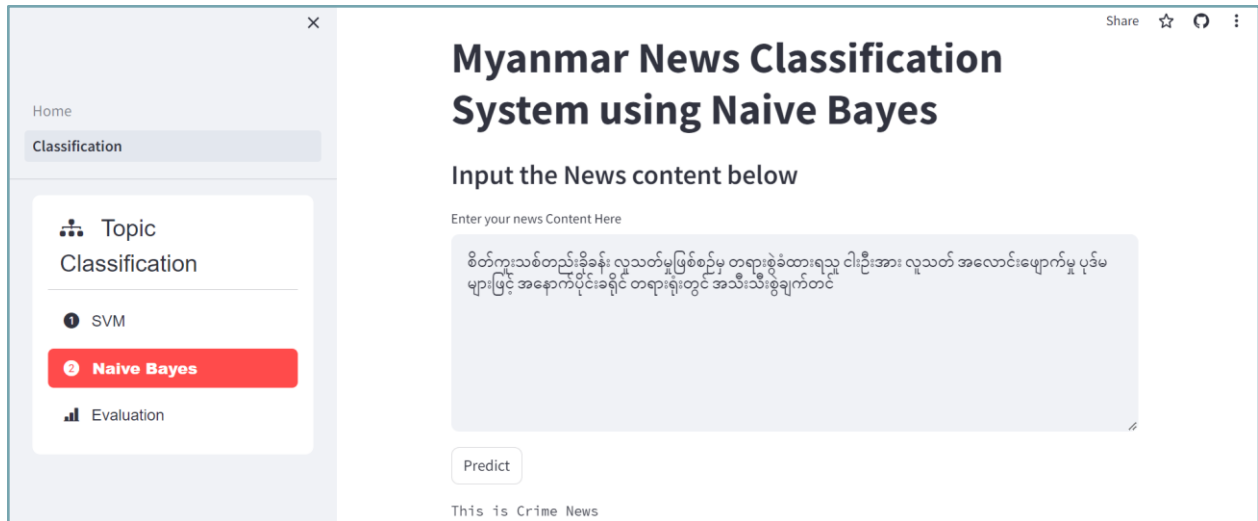


Figure 6.3 Myanmar News Naive Bayes Classifier's Interface

At the end, users will be able to see the "Experimental Results for Two Classifiers" and "Accuracy Comparison with Different Models" under the evaluation page. Users can access the evaluation page in the navigation section.

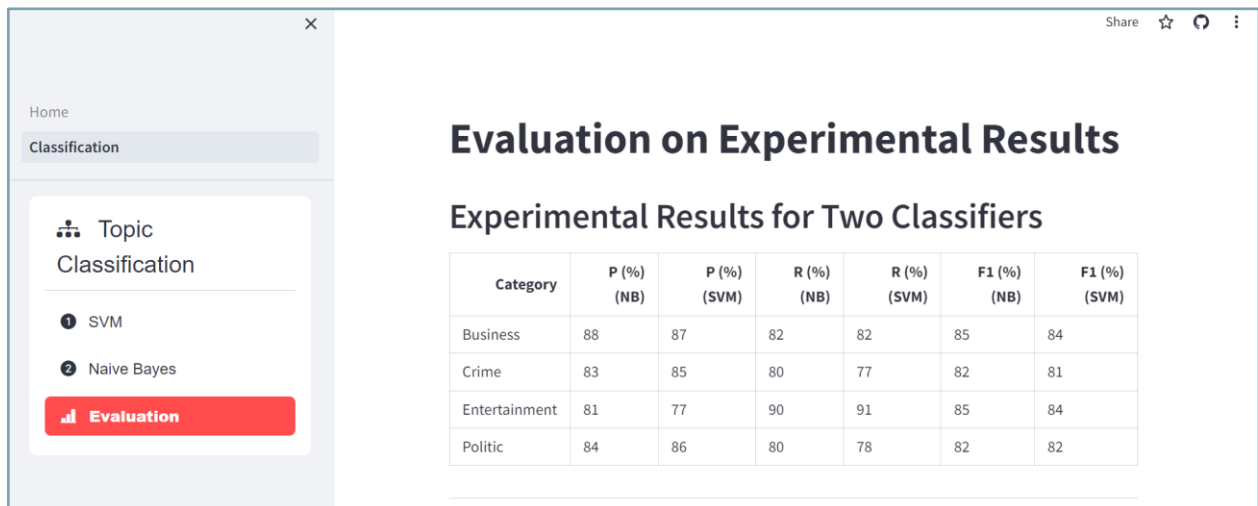


Figure 6.4 Evaluation Interface of Myanmar News Classification

6.3 Myanmar News Summarization

For the summarization part, there are three pages included. They are Extractive Summarization page, Monolingual Abstractive Summarization page, and Multilingual Abstractive Summarization page. Each page offers a unique summarization process. The extractive page

performs the extractive-wise summarization process while the Monolingual and Multilingual pages perform the abstractive-wise summarization.

In the extractive summarization page, user can enter their desired article in the "Input Area" text box. Then, the user can choose the number of sentences that the user wishes to generate by using the slider. When the user is ready, he can click the "Summarize!" button to generate the summarized article. Below, the user can see the difference between the input article in the "Full Input" box (left) and the generated summarization in the "Summarization" box (right).

The screenshot shows a web application interface for summarizing Myanmar news. On the left is a sidebar with navigation links: 'Home', 'Classification', and 'Extractive Summarization'. The main content area is titled 'Extractive Myanmar News Summarization'. It features an 'Input Area' with a text input field labeled 'Your Text' containing a paragraph of Myanmar text. Below the text field is a slider control labeled 'No. of sentences' with a range from 0 to 10, currently set at 2. A 'Summarize!' button is positioned below the slider. The interface is split into two side-by-side panels: 'Full Input' on the left and 'Summarization' on the right. Both panels display the same Myanmar text, allowing for a direct comparison between the original input and the generated summary.

Figure 6.5 Extractive Myanmar News Summarization's Interface

In both the Single Language (Monolingual) based summarization page and Multi Language (Multilingual) based summarization page, users can enter their desired article in the "Input Area" text box. When the user is ready, he can click the "Summarize!" button to generate the summarized article. Below, the user can see the difference between the input article in the "Full Input" box (left) and generated summarization in the "Summarization" box (right) in both pages.

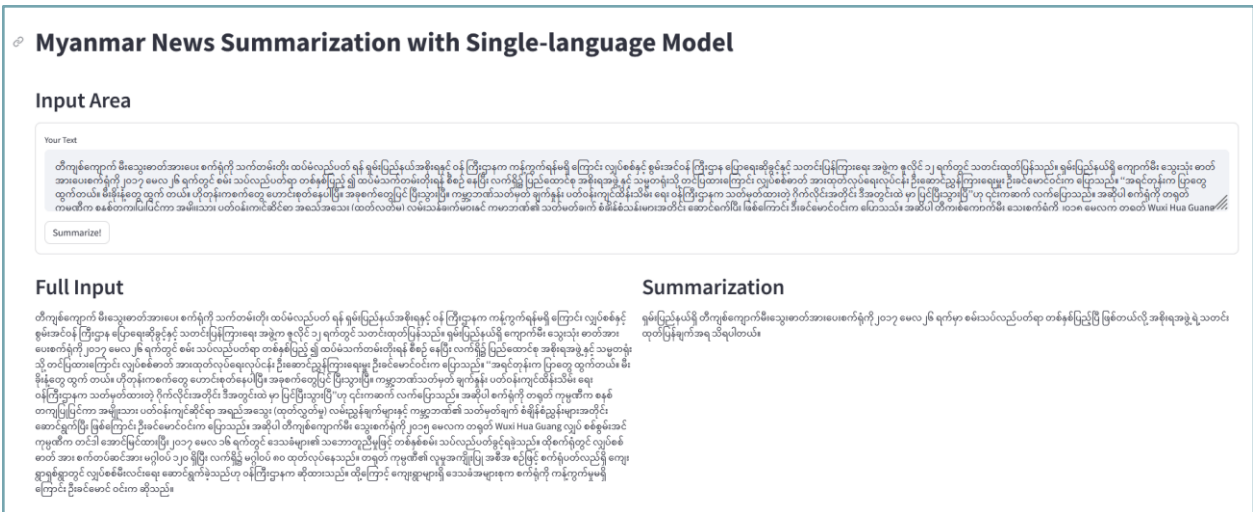


Figure 6.6 Monolingual Abstractive Myanmar News Summarization's Interface

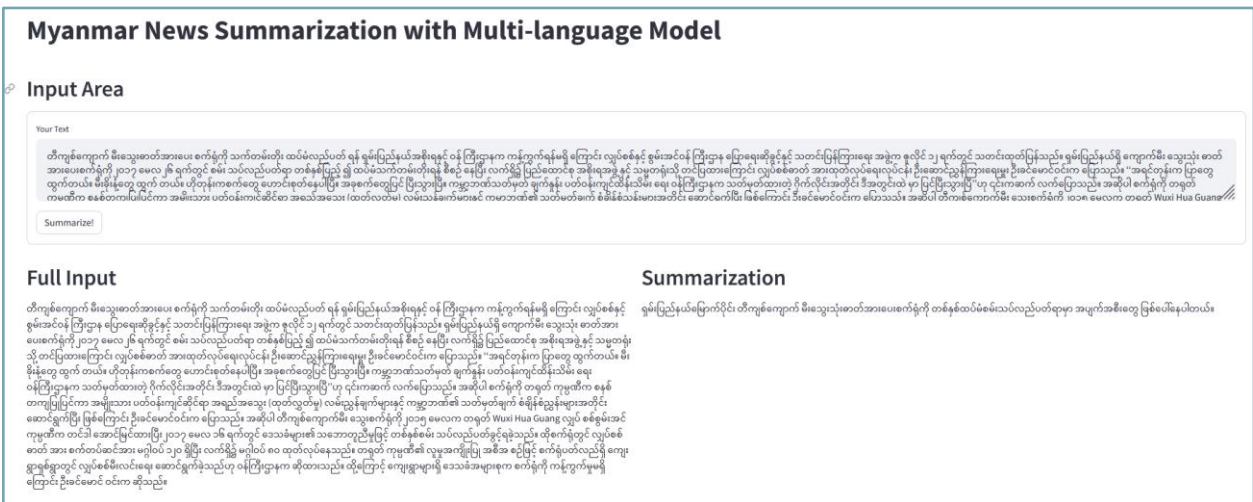


Figure 6.7 Multilingual Abstractive Myanmar News Summarization's Interface

Even though they have similar user interfaces, the model implementation process is different. For the Monolingual summarization page, the system utilized the Monolingual abstractive model and for the Multilingual summarization page, it utilized the Multi abstractive model.

Chapter 7 Conclusion

7.1 Conclusion

In conclusion, this project has explored and demonstrated the potential of Natural Language Processing (NLP) techniques in enhancing the analysis and understanding of Myanmar news content through classification and summarization. By leveraging machine learning algorithms and text processing methods have made significant strides in addressing the challenges associated with the vast amount of textual data available in the Myanmar language.

Firstly, through text classification, now users can successfully categorize Myanmar news articles into distinct topics or categories such as politics, sports, entertainment, and business. Utilizing algorithms such as Multinomial Naive Bayes, Linear Support Vector Machine (SVM), and Random Forest, helped develop robust models capable of accurately classifying news articles based on their content. This classification framework can benefit in many areas such as personalized news articles based on user preferences can be done. This ensures that users receive content aligned with their interests, enhancing the visions and goals of the news platforms.

Furthermore, text summarization aims to condense lengthy news articles into concise summaries while preserving the key information and meaning. Through extractive summarization techniques such as TF-IDF and Text Rank, successful summaries capturing the essence of original articles have been generated, facilitating quick comprehension of main points. Additionally, exploration into abstractive summarization, particularly through fine-tuning the mT5 model on the Myanmar language, has shown promising results. It was observed that fine-tuning the mT5 model specifically for Myanmar language text summarization outperformed using a multilingual finetune model. This underscores the importance of language-specific fine-tuning for achieving superior performance in abstractive summarization tasks.

The integration of classification and summarization techniques into NLP pipelines for Myanmar news analysis holds immense potential in various applications. From aiding journalists and researchers in information gathering and trend analysis to enhancing the accessibility of news content for the public, the advancements made here pave the way for a more comprehensive understanding and utilization of Myanmar language data.

7.2 Future Work

While significant progress has been made in this project on enhancing NLP for Myanmar news through classification and summarization, there are several avenues for future work and improvement that can further advance the field and address existing challenges.

Dataset Expansion and Annotation:

One of the critical factors in improving the performance of NLP models for Myanmar news analysis is the availability of high-quality annotated datasets. In the future, focusing on expanding existing datasets and annotating them with more diverse and comprehensive labels, allowing for more nuanced classification and summarization tasks. Collaborative efforts with industry partners, news organizations, and language experts can facilitate the creation of larger and more representative datasets.

Domain-specific Adaptation:

Myanmar news content spans various domains and topics, each with its unique vocabulary, terminology, and discourse patterns. In the future, investigating domain-specific adaptation techniques to tailor NLP models for specific topics such as technology, finance, or health. Domain-specific language models and annotated datasets can improve the accuracy and relevance of classification and summarization results for different news categories.

Fine-tuning Models for Myanmar Language:

While both extractive summarization and abstractive summarization methods have shown promise in condensing news articles, future research should explore more advanced abstractive summarization techniques. Exploring alternative pre-trained models beyond mT5 could enhance summarization performance. This could involve utilizing models like BERT, GPT, or XLNet, and investigating ensemble techniques for more comprehensive summarizations.

User-centric Applications and Interfaces:

Ultimately, the success of NLP research lies in its real-world applications and impact on end-users. In the future, prioritizing the development of user-centric NLP applications and interfaces that cater to the needs and preferences of Myanmar news consumers, journalists, researchers, and policymakers. User studies and feedback mechanisms can inform the design and refinement of NLP tools for improved usability and utility.

REFERENCES

- [1] T.T.Zaw | K.M.Soe - Myanmar Text Classifier Using Genetic Algorithm - National Journal of Parallel and Soft Computing, Volume 01, Issue 01, March-2019
- [2] K.T.Nwet | A.H.Khine | K.M.Soe - Automatic Myanmar News Classification - 15th International Conference On Computer Applications (ICCA2017)
- [3] S.S.Lwin, K.T.Nwet - Myanmar news summarization using different word representations - International Journal of Electrical and Computer Engineering (IJECE) Vol. 11, No. 3, June 2021
- [4] Y. Thu and W. P. Pa, "Generating Myanmar News Headlines using Recursive Neural Network," 2020 IEEE Conference on Computer Applications(ICCA), Yangon, Myanmar, 2020, pp. 1-6.
- [5] K.T.Nwet - MACHINE LEARNING ALGORITHMS FOR MYANMAR NEWS CLASSIFICATION - International Journal on Natural Language Computing (IJNLC) Vol.8, No.4, August 2019
- [6] S.S.Lwin | K.T.Nwet - Extractive Summarization for Myanmar Language - 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)
- [7] T.Yu | K.T.Nwet - Myanmar News Sentiment Analyzer using Support Vector Machine Algorithm - International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No.6, November – December 2019
- [8] T.Hasan | A.Bhattacharjee | M.S.Islam | K.Mubasshir | Y.F.Li | Y.B.Kang | M.S.Rahman | R.Shahriyar - 2021. XL-Sum: Large-Scale Multilingual Abstractive Summarization for 44 Languages. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4693–4703, Online. Association for Computational Linguistics.
- [9] Farahani | Mehrdad | Gharachorloo | Mohammad | Manthouri | Mohammad - Leveraging ParsBERT and Pretrained mT5 for Persian Abstractive Text Summarization. 1-6. 10.1109/CSICC52343.2021.9420563.

[10] Taunk, D. | Varma, V - Summarizing indian languages using multilingual transformers based models. arXiv preprint arXiv:2303.16657.

[11] Myanmar News Dataset : <https://github.com/kyawswa/Myanmar-News-Data>

[12] XL-SUM Dataset : <https://huggingface.co/datasets/csebuetnlp/xlsum>

[14] Pydiaungsu Library : <https://github.com/kaunghtetsan275/pyidaungsu/tree/master>

Appendix

Myanmar News Dataset

Index	News	Category	category_id
1	တီကျစ်ကျောက် မီးသွေးဓာတ်အားပေး စက်ရုံကို သက်တမ...	business	0
2	ရန်ကုန်တိုင်းအစိုးရ အစီအစဉ်ဖြင့် ရောင်းချမည့် ...	business	0
3	ရန်ကုန်မြို့သစ်စီမံကိန်း အတွက် လမ်းကြောင်းကွန်...	business	0
4	ပြည်တွင်း စက်သုံးဆီဈေး ရက်ပိုင်းအတွင်း ပြန်လည်...	business	0
5	ကချင်ပြည်နယ် မြစ် ကြီးနားနှင့် ချီဖွေမြို့အတွက်...	business	0

XL-SUM Dataset from Hugging Face

Datasets: csebuetnlp/xlsum like 80 Dataset card Viewer Files and versions Community

Subset (45)
burmese · 5.71k rows

Split (3)
train · 4.57k rows

Search this dataset

id	url	title	summary	text
string · Lengths	string · Lengths	string · Lengths	string · Lengths	string · Lengths
8	36	11	8	285
62	267	208	924	27.7k
151283_syria_uk_airstrikes	https://www.bbc.com/burmese/world/2015/12/151283_syria_uk_airstrikes	စီရီးယား IS တွေကို ယူကရိန်းကောင်စီက ရိုက်ခတ်	မြို့တို့သို့ တော်ဝင် လေတပ်မှ တိုခိုက်မှုများ တိုက်လေယာဉ်တွေက ဆို...	IS ပိုင် နေရာတွင်တွေ့ရှိ မြို့တို့သို့ တိုက်လေယာဉ် ၄ စီးက...
150908_turkey_strikes_kurdish_pkk_iraq	https://www.bbc.com/burmese/world/2015/09/150908_turkey_strikes_kurdish_pkk_iraq	အီရတ် ကာဘန် သူရဲကောင်းတို့ တွေ့ရှိ ဖမ်းရန်	တူရကီ အစိုးရက လူနဲ့အတူ တပ်မှန်တွေ တိုက်ခိုက် ခဲ့ရန်...	အီရတ် ၁၆ နဲ့ အီရတ် ၄ တိုက်လေယာဉ်တွေက အီရတ် ဖြတ်...
141222_nkorea_uncs	https://www.bbc.com/burmese/world/2014/12/141222_nkorea_uncs	မြောက်ကိုရီးယားလူမျိုးစုက တောင်ကိုရီးယား မှ	မြောက်ကိုရီးယား လူအစုအဖွဲ့အစုအဖွဲ့အစု ကို တည်ထောင်...	မြောက်ကိုရီးယား လူအစု ကို ဖမ်းဆီး တွေ့ ကျူးလွန်နေ...
140403_ilo_labour_rep	https://www.bbc.com/burmese/burma/2014/04/140403_ilo_labour_rep	အလုပ်သမား ကိုယ်စားလှယ် ရွေးချယ်ရေး အဖွဲ့က	လေ့ကျင့်မှု နိုင်ငံတကာ အလုပ်သမား အဖွဲ့အစည်း ILO က...	အလုပ်သမား ညီလာခံ ကိုယ်စားလှယ် အဖွဲ့က အစီ...
130619_india_floods	https://www.bbc.com/burmese/world/2013/06/130619_india_floods	အိန္ဒိယ မြောက်ပိုင်း ချီသုတ်မြို့ ရေလွှဲမှု	အိန္ဒိယ နိုင်ငံမြောက်ပိုင်းမှာ မိုး သွန်းထား ခဲ့တာကြောင့် ရေလွှဲ...	အိန္ဒိယမှာ ချီသုတ်မြို့ ရေလွှဲ ကြီးမား ပျက်စီးမှု ဖြစ်ပေါ်...
141010_malala_satyarthi_nobel	https://www.bbc.com/burmese/world/2014/10/141010_malala_satyarthi_nobel	မာလာလာနှင့် ဆတ်တီယာသီ နိုဘယ် ညွှန်ကြားရေးအရာရှိ	၂၀၁၄ ခုနှစ် အတွက် နိုဘယ် ညွှန်ကြားရေး အကဲဖြတ် ကော်မတီ...	မာလာလာနှင့် ဆတ်တီယာသီ နိုဘယ် ညွှန်ကြားရေး အကဲဖြတ် ကော်မတီ...

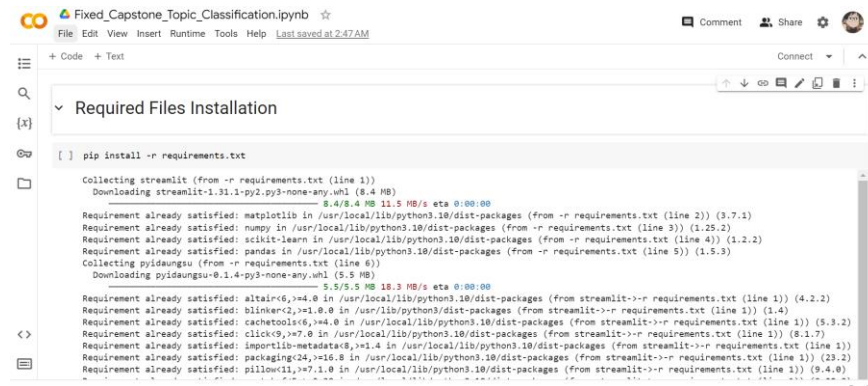
Programming Language Use

Use Python as programming Language with version 3.12.



Implementation Environment

Use Google Colab for implementation environment.



```
Fixed_Capstone_Topic_Classification.ipynb
File Edit View Insert Runtime Tools Help Last saved at 2:47 AM

+ Code + Text

Required Files Installation

[ ] pip install -r requirements.txt

Collecting streamlit (from -r requirements.txt (line 1))
  Downloading streamlit-1.31.1-py2.py3-none-any.whl (8.4 MB)
    9.4/8.4 MB 11.5 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 2)) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 3)) (1.25.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 4)) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 5)) (1.5.3)
Collecting pydaungsu (from -r requirements.txt (line 6))
  Downloading pydaungsu-0.1.4-py3-none-any.whl (5.5 MB)
    5.5/5.5 MB 18.3 MB/s eta 0:00:00
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit->-r requirements.txt (line 1)) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/lib/python3/dist-packages (from streamlit->-r requirements.txt (line 1)) (1.4)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit->-r requirements.txt (line 1)) (5.3.2)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit->-r requirements.txt (line 1)) (8.1.7)
Requirement already satisfied: importlib-metadata<8,>=4 in /usr/local/lib/python3.10/dist-packages (from streamlit->-r requirements.txt (line 1)) (7.0.0)
Requirement already satisfied: packaging<24,>=16.8 in /usr/local/lib/python3.10/dist-packages (from streamlit->-r requirements.txt (line 1)) (23.2)
Requirement already satisfied: pillow<11,>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit->-r requirements.txt (line 1)) (9.4.0)
```

Application deployment Environment

Use Streamlit Cloud for deployment environment.

