TRIBHUVAN UNIVERSITY

Prime College

Nayabazar, Kathmandu, Nepal

A Project Report

On

"Snap News:
News Classifier and Summarizer"

Submitted By:

Anuj Shrestha (20471/075)

Bibek Thapa (20473/075)

Yash Maharjan (20508/075)

A Project Report Submitted in partial fulfillment of the requirement of **Bachelor of Science in Computer Science & Information Technology (BSc.CSIT) 7th Semester** of Tribhuvan University, Nepal

Baisakh, 2080

Snap News

[CSC 412]

A project report submitted for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer science & Information Technology awarded by Tribhuvan University.

Submitted By

Anuj Shrestha (20471/075)

Bibek Thapa (20473/075)

Yash Maharjan (20508/075)

Submitted To

Prime College

Department of Computer science

Affiliated to Tribhuvan University

Khusibun, Nayabazar, Kathmandu

Baisakh, 2080

Date: 31th Baisakh, 2080

SUPERVISOR'S RECOMMENDATION

It is my pleasure to recommend that a report on "**Snap News: News Classifier and Summarizer**" has been prepared under my supervision by **Anuj Shrestha**, **Bibek Thapa** and **Yash Maharjan** in partial fulfillment of the requirement of the degree of Bachelor of Science in Computer Science and Information Technology (BSc.CSIT). Their report is satisfactory to process for the future evaluation.

……

Mr. Sravan Ghimire

Supervisor

Department of Computer Science & IT

Prime College

Date: 31th Baisakh, 2080

CERTIFICATE OF APPROVAL

The undersigned certify that he has read and recommended to the Department of Computer Science and Information Technology for acceptance of report entitled "**Snap News: News Classifier and Summarizer**" submitted by **Anuj Shrestha, Bibek Thapa** and **Yash Maharjan** in partial fulfillment for the degree of Bachelor of Science in Computer Science and Information Technology (BSc.CSIT), Institute of Science and Technology, Tribhuvan University.

…….	……
Mr. Narayan Prasad Sharma	**Ms. Rolisha Sthapit**
Principal	**Program Coordinator**
……	……
Mr. Sravan Ghimire	**Mr. Jagdish Bhatta**
Supervisor	**External Examiner**

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our project supervisor, **Mr. Sravan Ghimire**, in whose contribution stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report. Furthermore, we would also like to acknowledge with much appreciation the crucial role of the staff of Prime College, who gave the permission to use all required equipment and the necessary materials to complete the task. We are thankful and fortunate enough to get constant support from our seniors and every teaching staff of B.Sc. CSIT Department which helped us successfully complete our project. We would also like to extend our regards to all the non-teaching staff of CSIT department for their timely support. We have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices. Our appreciations also go to each and every one of our colleagues for their encouragement and support in developing the project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere appreciation to all of them.

With respect,

Anuj Shrestha (20471/075)

Bibek Thapa (20473/075)

Yash Maharjan (20508/075)

ABSTRACT

The Snap News project is a tool to categorize, summarize and scrape the news articles of either Nepali or English language. This project categorizes news articles, and provide a quick summary of the news article thus entered to make it easier for the people to access and understand. It uses Natural Language processing technology to identify the category of the news hence making the reader easier to access the news of their choice. It also provides a summary of the news text using the extractive summary process in which the application identifies the important sentences in the news articles and summarize the long text into shorter readable form. It also consists of a news scraper that can scrape the news off the news sites and then provide a central location for the news reading. The scraped news can either be of Nepali or English. This project can be used by many users such as journalists, news enthusiasts, etc. Journalists can use this tool to find the information quickly and then categorize the news stories into respective categories and the normal users can use this to get quick information of the current news and then get updated.

KEYWORDS: *News, Natural Language Processing, Machine Learning, Classify, Summarize, Scrape*

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
BBC	British Broadcasting Company
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
CSS	Cascading Style Sheet
CSV	Comma Separated Values
GRU	Gated Recurrent Unit
GPT	Generative Pre-trained Transformer
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IDF	Inverse Document Frequency
JS	JavaScript
KNN	K-Nearest Neighbor
LR	Logistic Regression
LCS	Longest Common Subsequence
ML	Machine Learning
MUI	Material User Interface
MNB	Multinomial Naive Bayes
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
RAM	Random Access Memory

RBF	Radial Basis Function
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SVM	Support Vector Machine
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TXT	Text
UI	User Interface
UML	Unified Modelling Language
URL	Uniform Resource Locator

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Introduction

Snap News is a web application that classifies and summarizes English and Nepali news articles. It also scrapes fresh news classifies them, summarizes them and presents them. It uses machine learning model to classify news. For summarization TF-IDF Matrix is used. Lastly Beautiful Soup is used to scrape news. Snap News make use of Natural Language Processing.

In this digital age everyone seems to have a short attention span. Everyone wants things fast and short. The goal of this project is to solve this very problem. Snap News provides a summarized news from news sources Ekantipur, SetoPati, BBC and Kathmandu Post. It also summarizes them and latest news is shown.

Snap News also provides a form like interface. Here you can enter any news in language Nepali and English, select number of sentences to summarize to and classify the news.

1.2 Problem Statement

In this digital age people tend to have a short attention span. People only want the important stuffs, entertaining stuffs or eye-catching stuffs. They don't sit around to read long versions of news article. Most casual readers just go through the headings. But sometimes heading might not be enough. So, what could be done? One solution might be to develop a news categorization and summarization model.

Snap News is designed to scrape for news in news websites. It then categorizes them. Making use of NLP and TF-IDF Matrix the most important sentences of a news article is selected and presented. The categorization of news could be used to target specific user with specific tastes. This is used for personalization. The summary feature of Snap News is then used to extract useful information about news article. So, user don't miss out on news and at the same time get well informed from reading a small chunk of a longer news article.

1.3 Objectives

The project aims to meet the following objectives:

1. To develop a machine learning model to classify English and Nepali news into category business, entertainment, politics, sports and tech.
2. To implement an extractive summarization algorithm that can identify and extract the most important sentences from English and Nepali news article.
3. To implement a web scraper that can collect news articles from multiple sources, classify and summarize them.

1.4 Scope and Limitation

The scope of the project is to develop a news classification, summarization, and scraping system that can provide users with timely and accurate news articles based on their preferences. The system will use machine learning algorithms to classify news articles into different categories and an extractive summarization algorithm to provide short, accurate summaries of each article. The system will also have a news scraper component that can extract news articles from various sources in real-time.

The limitations of the project may include:

- Accuracy: Since the project uses machine learning algorithms the project may not provide the accurate information at the beginning. As there would be limited data sets.
- Language: Since the project can only classify, summarize and scrape the news in either Nepali or English it wouldn't be able to handle other languages.
- Subjectivity: The accuracy of the summarization algorithm may be affected by the subjectivity of the news article. Some news articles may contain subjective information, making it difficult for the algorithm to accurately identify the most important sentences.

1.5 Development Methodology

The development methodology used for the Snap News project is incremental delivery, which is a kind of iterative software development process. The project is divided into a series of incremental iterations. Each function or features are being added upon the previous one to gradually deliver a full-fledged functional system.

The incremental delivery approach is particularly suited for projects which involves a high level of complexity, or where the requirements are not clearly-defined or may change over the period of time. Breaking the project down into smaller iterations, allows for a more manageable and flexible development process. Also, it allows for feedback and corrections to be made throughout the process rather than waiting until the end of the project to evaluate the final product. For the Snap News project, the first iteration focuses on building the basic infrastructure of the system, including the installation of web scraping functionality to collect news articles from various sources and setting up of initial data processing pipelines for classification and summarization. Subsequent iterations then focus on adding more advanced features, such as improving the accuracy of classification algorithms and incorporating more complex summary techniques. Before going on to the next iteration, each iteration was rigorously tested, ensuring that the system was reliable and usable throughout the whole development process.

Overall, the incremental delivery technique proved to be a successful strategy for creating the Snap News project, enabling a flexible and adaptive development process that finally resulted in the successful delivery of a useful and functional system.

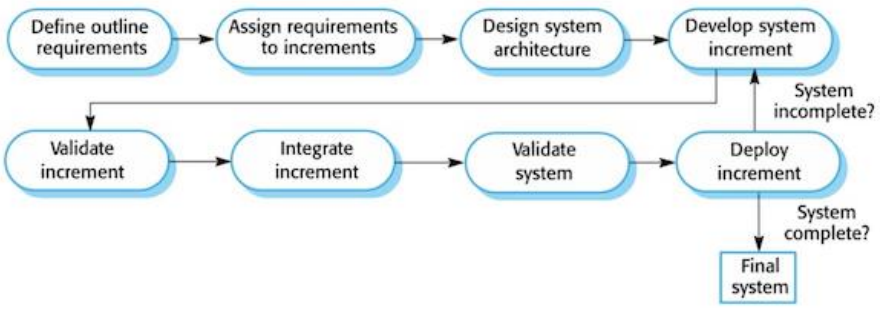


Figure 1.1: Incremental Delivery [1].

1.6 Report Organization

This report is organized in 6 chapters. The first chapter consists of the introduction of overall project which is again further categorized into 6 subchapters. This subchapter consists of the problem statement, objectives, scope and limitations and development methodology of the project. The second chapter consists background study of fundamental theories, general concepts and literature review i.e., review of similar projects, theories and results by other researchers. The third chapter is analysis part which includes system analysis, requirement analysis and feasibility analysis. The fourth chapter includes overall design of the system. The fifth chapter discuses about the implementation and testing of the system using different related tools and the final chapter consists of conclusion and future recommendation regarding the current system.

CHAPTER 2
BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

News summarization and classification are two important fields of study in the realm of natural language processing (NLP) and machine learning (ML). Various techniques have been researched and developed to improve the accuracy and efficiency of these systems.

Classification of news article using machine learning model requires various steps. First step is always to collect news articles and properly label them. The news article is tokenized. Tokenized data preprocessing like removing stop words, numbers, special characterization are done. Lemmatization and Stemming are performed to further process the data. Finally, the textual form of data is represented to numerical format using TF-IDF values or word embedding. A proper machine learning model is then selected. Possible learning models being Naive Bayes, logistic regression, support vector machines, decision trees, or deep learning models such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs). Model is then trained. After training the model, its performance can be evaluated using various evaluation metrices.

Extractive Summarization is summarizing a paragraph or group of sentences such that no new sentences are generated and the final summary is sentences already present in paragraph. Extractive Summary puts together already present sentences that seems to have a greater importance value. Here paragraphs are preprocessed by removing numerical data and special characters. Lemmatization and Stemming are performed while stop words are not removed. Each Sentence are given a TF-IDF values. Then the sentences are sorted in accordance to TF-IDF values. Most important sentences have greater TF-IDF value. Then summarized sentence are displayed on user choice of sentences.

Thus, Classification and Summarization are done making proper use of NLP and ML.

2.2 Literature Review

Text classification is an essential task in natural language processing that supports various applications such as sentiment analysis, topic classification, and question-answering systems. Early research in this field relied on manual rules and knowledge engineering methods, which required significant manpower, material wealth, and financial support. However, with the emergence of machine learning, text categorization has been gradually replaced by automatic classification by machines. Traditional machine learning algorithms such as SVM, KNN, MNB, LR, and decision tree are commonly used in text classification. However, these models require manually designed features, which are highly subjective and task-specific, and can only extract shallow features, leading to overfitting and inefficiency in handling large amounts of data.

One popular method for text classification is to use the term frequency-inverse document frequency (TFIDF) matrix, which represents the frequency of each term in a document and its inverse document frequency. Support vector machines (SVMs) are a popular machine learning algorithm for text classification.

There have been several studies that have used SVM with TFIDF matrix for text classification. In a study by Joachims (1998), the author used Naïve Bayes, Rocchio, C4.5, KNN, SVM (poly) and SVM (rbf) for text classification and achieved high accuracy on several benchmark datasets. The performance measure metrics was precision/recall breakeven point. Here SVM (rbf) performed well with micro average score of 86.4. Naïve Bayes, Rocchio, C4.5, KNN and SCM (poly) scored scores 72.0, 79.9, 79.4, 82.3 and 86 respectively. Thus, SVM (rbf) performed the best [2]. It shows SVM leads when it comes to accurately classifying texts.

Another study by Krishnalal et al. (2010) used Hidden Markov Model SVM, SVM and KNN for web news classification. The class were Sports, Finance and Politics. KNN accurately predicted Sports, Finance and Politics with accuracy 83.25%, 80.22% and 82.26%. SVM gave accuracy of 87.67%, 82.57% and 86.55% respectively. Hidden Markov Model SVM gave best accuracy of 92.45%, 96.34% and 90.76% respectively [3]. It now becomes more evident that SVM works great for multiclass text classification.

The fact that SVM works better than other models such as K-NN, Rocchio, Bayes for high dimensional data is supported by paper written by Gharib et al. (2010). The study included

class of Arts, Economics, Politics, Sports, Woman and Information Technology. This collection consisted of 1,132 documents, 95138 words (22347 unique words). SVM performed the best [4] . Although training time for SVM was slower compared to other. According to study by Raghuveer et al. (2007) on Indian languages SVM showed the best result when compared to Naïve Bayes and KNN. The categories were Politics, Business, Sports and Cinema. The study was done in ten language corpuses. SVM outperformed Naïve Bayes and KNN with f1-score of 78.63% when compared to 68.9% and 66.31% respectively [5].

In conclusion, SVM, KNN or MNB with TFIDF matrix has been successfully used for news text multiclass classification in several studies. The combination of these models with TFIDF matrix has shown to achieve high accuracy for multiclass classification of news articles in different languages, including Arabic, English, Marathi, and Korean.

CHAPTER 3
SYSTEM ANALYSIS

3.1 System Analysis

Systems analysis is a process of studying a system or organization in order to understand its components, how they interact and how they can be improved. It is a holistic approach that looks at the system as a whole and identifies the relationships between its parts. The goal of systems analysis is to identify problems and inefficiencies in the current system and to propose solutions for improvement.

3.1.1 Requirement Analysis

Requirement analysis is a critical phase in software development that involves gathering, analyzing, and documenting the needs and constraints of the project. For the Snap News project, the requirement analysis process aimed to identify the key features and functionality of the system, as well as any constraints or limitations that needed to be considered during development.

3.1.1.1 Functional Requirements

Functional requirements aim to provide the overview of how the system works. Here, the functionalities such as services, tasks and functions required for the system is shown. The functional requirements are: -

- **News Classification and Summarization**
  The system should be able to classify news articles into relevant categories such as politics, sports, entertainment, and so on and the system should be able to generate summary snippets for each news article. The process starts when a user enters the news they want to classify, then enter the summarization count which is the number of sentences they want as a summary. If the sentence count is greater than the sentences of whole news text it'd throw an error, else it'd use the model to predict the category of the news and the summarized news to the user.

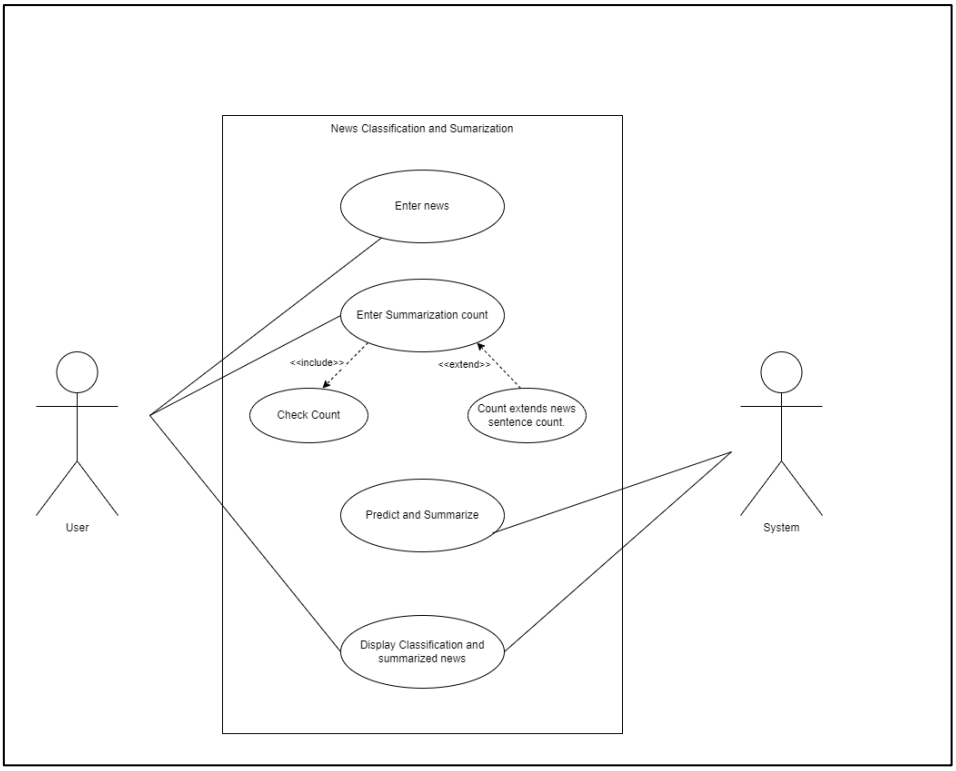


Figure 3.1: Use Case for News Classification and Summarization

- **News Scraping**

  The system should be able to scrape news articles from various online news sources. The process starts when the user selects the language and the category of the news, they want to view the scraper then fetches the news for the user and as well as summarize the content of the news.

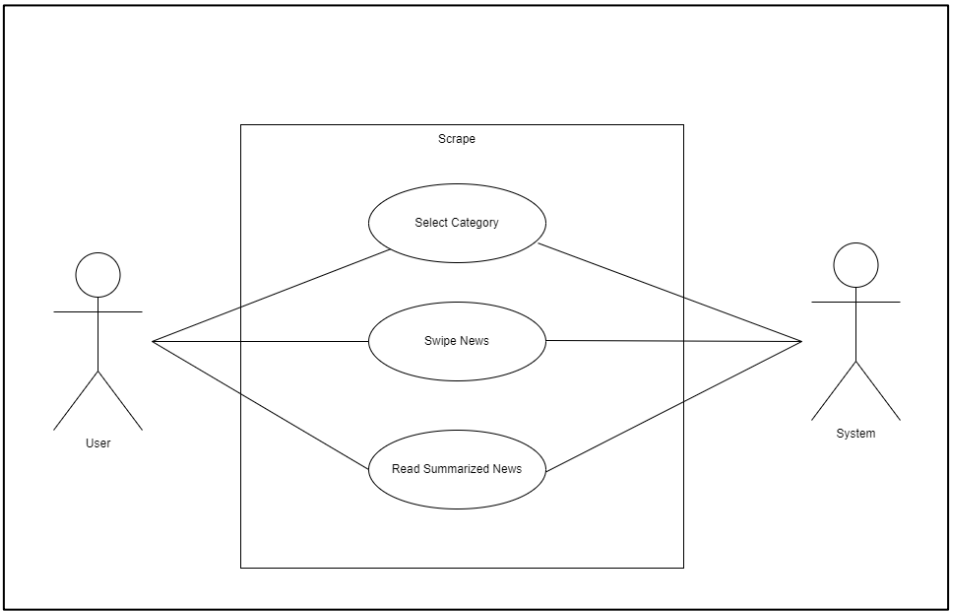


Figure 3.2: Use Case for News Scraping

3.1.1.2 Non-Functional Requirements

The points below focus on the non-functional requirement of the system:

- Performance: The system should be highly performant and responsive, with minimal lag or delay when loading and displaying news articles.
- Usability: The system should be easy to use and navigate, with a clear and intuitive interface that appeals to a wide range of users.
- Scalability: The system should be scalable and able to handle large volumes of traffic and data, as the user base grows over time.
- Availability: The system should be highly available and able to handle high volumes of traffic without downtime or interruption, to ensure a smooth user experience.

3.1.2 Feasibility Analysis

3.1.2.1 Technical Feasibility

The technical feasibility of the project is high, as the required hardware and software resources are widely available and accessible. Requirements of our system can be categorized as:

Hardware Requirements:

- A computer with a minimum of 4 GB of RAM and a multi-core processor
- Adequate storage space to store news articles and system files
- A stable and fast internet connection to enable web scraping

Software Requirements:

- Operating System: Windows 10, Linux, or MacOS
- Web Browser: Chrome, Firefox, or Safari
- Python 3.6 or above: This is required to run the Flask web framework and the various Python libraries used in the project, such as BeautifulSoup for web scraping and NLTK for natural language processing.
- Flask web framework: This is required to build the web application and API for the system.

- ReactJS: This is required to build the frontend of the web application and to create a responsive user interface.
- Git and GitHub: These are required for version control and collaboration among the development team.

In addition to the above requirements, some optional software may be used for development such as virtual environments like Anaconda or Virtualenv to manage dependencies.

3.1.2.2 Operational Feasibility

The operational feasibility of the project is moderate, as the project requires the development of a reliable and accurate news article scraper, classifier, and summarizer. The success of the project depends on the ability of the system to collect and analyze news articles from various sources accurately and efficiently. The system will also need to be maintained and updated regularly to ensure it remains current and relevant.

3.1.2.3 Economic Feasibility

The economic feasibility of the project is also high, as the project does not require any significant upfront investment or ongoing operational costs. The required hardware and software resources are typically available on most computers and can be acquired at a relatively low cost. In addition, the development can use open-source software and libraries to reduce the cost of development.

3.1.2.4 Schedule Feasibility

The time given for the completion of this project was a whole semester. So, the project had enough time for completion. Since, the project has some machine learning mechanisms the project took some more time than done usually. Hence, the project has been developed according to the following time schedule to make our application schedule feasible:

Table 3.1: Schedule Table

S. No	Task Name	Duration	Start Date	End Date
1	Planning	4 Days	Nov 25, 2022	Nov 30, 2022
2	Analysis	6 Days	Dec 01, 2022	Dec 08, 2022
3	Design	19 Days	Dec 09, 2022	Jan 04, 2023
4	Coding	20 Days	Jan 04, 2022	Jan 31, 2023
5	Testing	5 Days	Jan 31, 2022	Feb 06, 2023
6	Implementation	6 Days	Feb 06, 2023	Feb 13, 2023
7	Documentation	80 Days	Nov 25, 2022	Feb 13, 2023

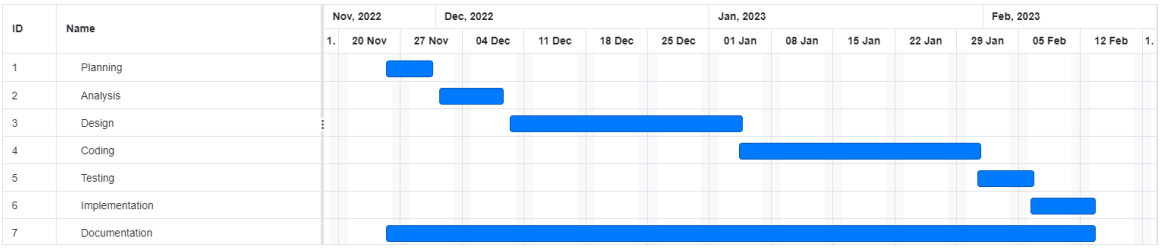


Figure 3.3: Gantt Chart

3.1.3 Analysis

3.1.3.1 Object Modeling using class diagram

Class diagram in the UML is a type of static structure diagram that describes the structure of a system, it shows system's classes along with their attributes, operations and relationships among objects.

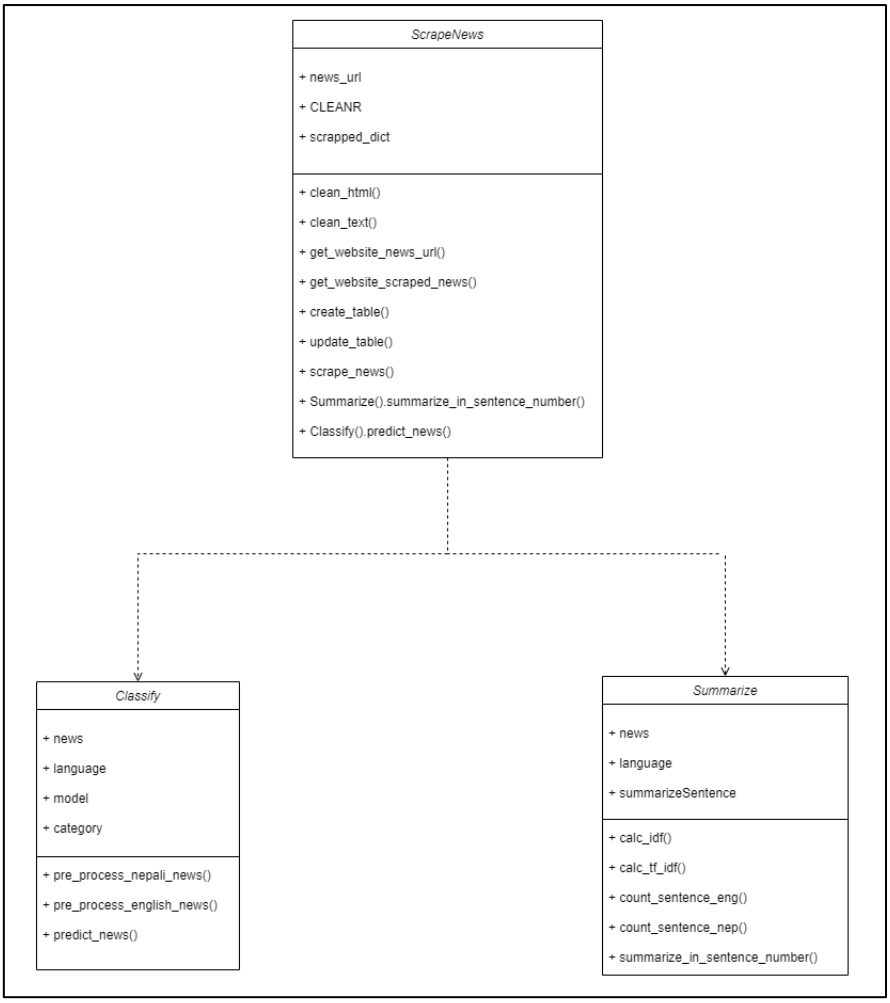


Figure 3.4: Class Diagram

In the above class diagram Figure 3.4, there consists of three classes namely ScrapeNews, Classify and Summarize. The classify class consists of the attributes of news, language, model and category. Another class, The Summarize class consists of attributes of news, language and summarizeSentence. Finally, the ScrapeNews class consists of attributes news_url, CLEANR and scrapped_dict. The news attribute is the news given as an input to the model, language is the attribute which detects whether the news input is in Nepali or English. The category attribute defines the category of the news article. The summarizeSentence is the attribute which is the number of the sentence the user wants the news to be summarized to. The news_url from the NewsScrapped Class consists of the url from which the news was retrieved, CLEANR is a regular expression used to clean the HTML content and scrapped_dict is a dictionary that stores the scrapped news articles.

The pre_process_nepali_news() and pre_process_english_news() methods are used to preprocess the news article for classification, while the predict_news() method predicts the

category of the news article based on the language, model, and preprocessed news article. he calc_idf() and calc_tf_idf() methods are used to calculate the importance of each word in the news article, while the count_sentence_eng() and count_sentence_nep() methods count the number of sentences in the English and Nepali news articles, respectively. The summarize_in_sentence_number() method generates a summary of the news article based on the number of sentences specified. The clean_html() and clean_text() methods are used to clean the HTML tags and other unwanted elements from the news article, while the get_website_news_url() and get_website_scrapped_news() methods are used to get the URLs of the news articles and the scrapped news articles, respectively. The create_table() and update_table() methods are used to create and update a database table for storing the news articles. The scrape_news() method scrapes news articles from the specified website and stores them in the database table. The Summarize () and Classify () are classes to summarize and classify the scraped news articles, respectively.

3.1.3.2 Dynamic Modeling using sequence diagram

Sequence diagram in the UML is a type diagram that illustrates the sequence of messages between objects in an interaction. It consists of a group of objects represented by lifelines and message they exchange during the interaction denoted by arrow symbols.

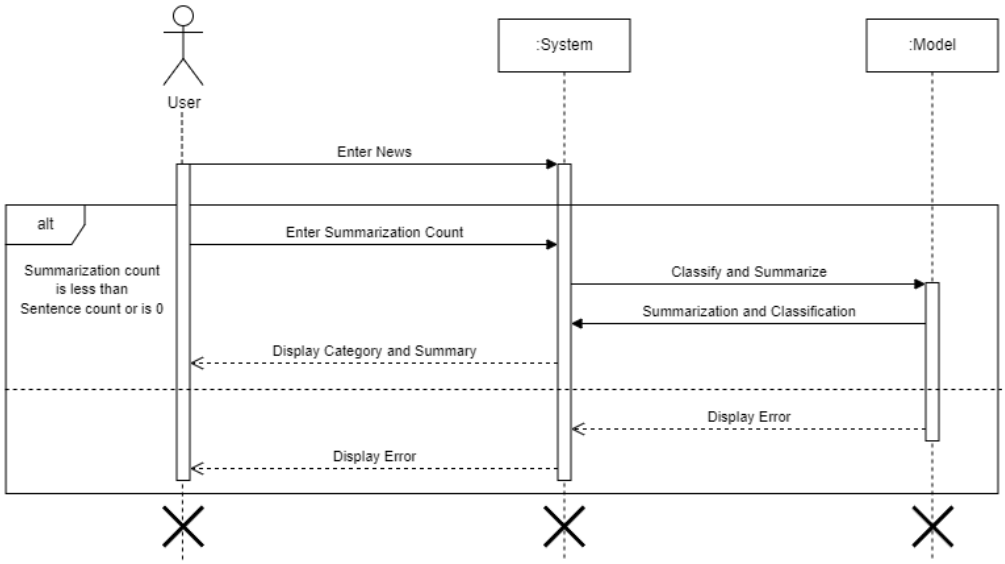


Figure 3.5: Sequence Diagram of Classification and Summarization

In the above Figure 3.5, the user first enters the news article they want to summarize or classify. After entering the news article in the form, the users input the numeric value of the summarization count which is the number count of the sentence the user wants as the summary. The system then uses the model to classify and summarize the news article. If the summarization count which is the user input number of sentence would be less than the number of sentences in the news article then the system would throw an error of the sentence count being less than the total sentence count of the news article. Else, it'd show the category of the news and the summary of the news article to the user.

3.1.3.3 Process Modeling using Activity Diagram

Activity diagram in the UML is a type of diagram that visually presents a series of actions or flow of control in a system. It shows workflows of stepwise activities with support for choice, iteration and concurrency.

The Figure 3.6 below shows the activity diagram of SnapNews. The user first inputs the news article and summarization count in the form-based input. The server then checks the sentence count of the news article and then the summarize count of the value input by the user. And, if the summarization count is less than sentence count and if the summarization count is not equal to zero. Then the model classifies the news article and summarizes it else the user would be given an error of the summarization count less than that of the sentence count or the value of the summarization count is equal to zero. Else, the news article would be classified into five categories of Business, Tech, Sports, Entertainment or Politics and the summary of the news article with the summarization count from the user would be shown to the user.

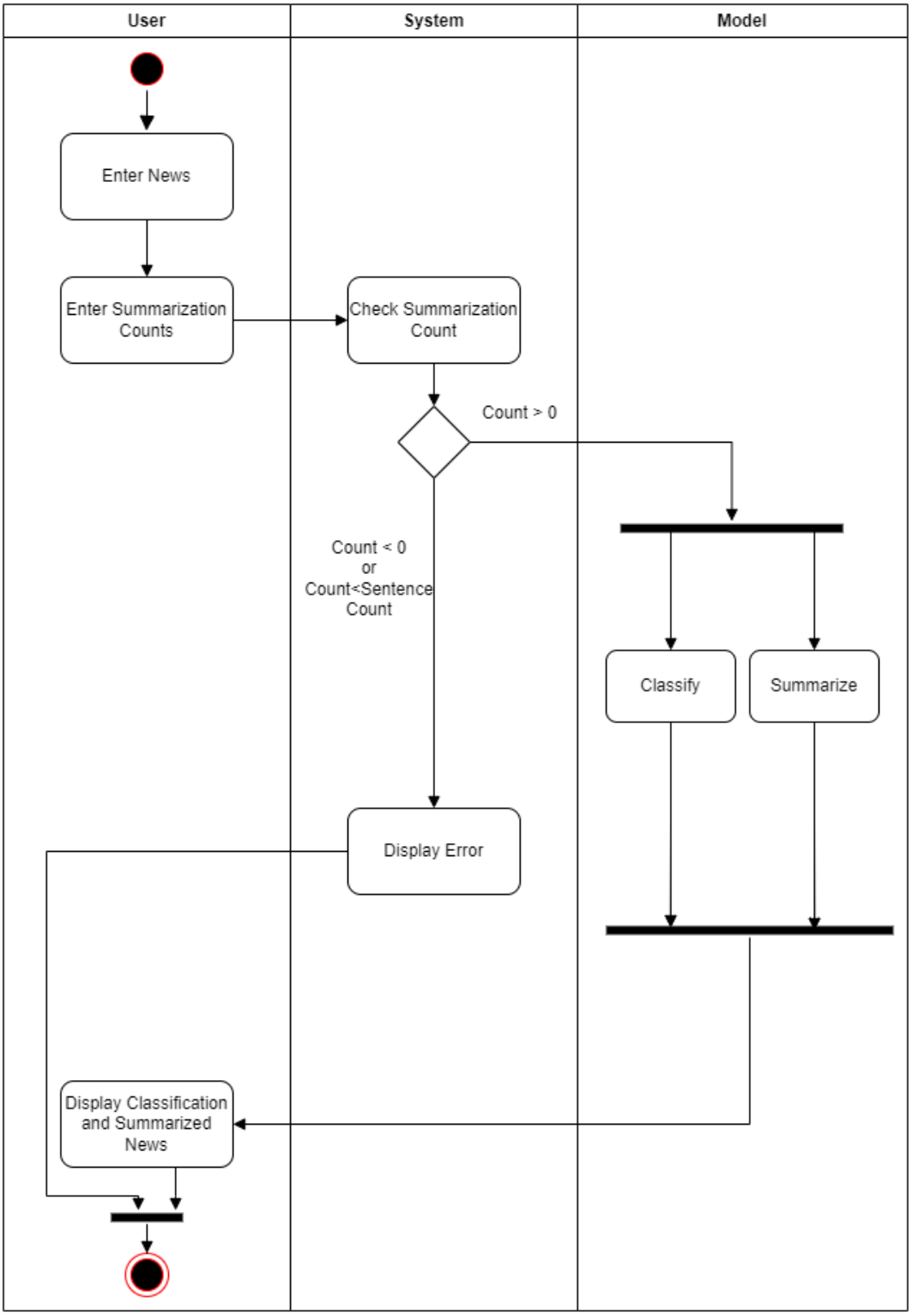


Figure 3.6: Activity Diagram of Classification and Summarization

CHAPTER 4
SYSTEM DESIGN

4.1 Design

System design is the process of representing architecture, interfaces, components that are included in the system. i.e., system design can be seen as the application of system theory to product development.

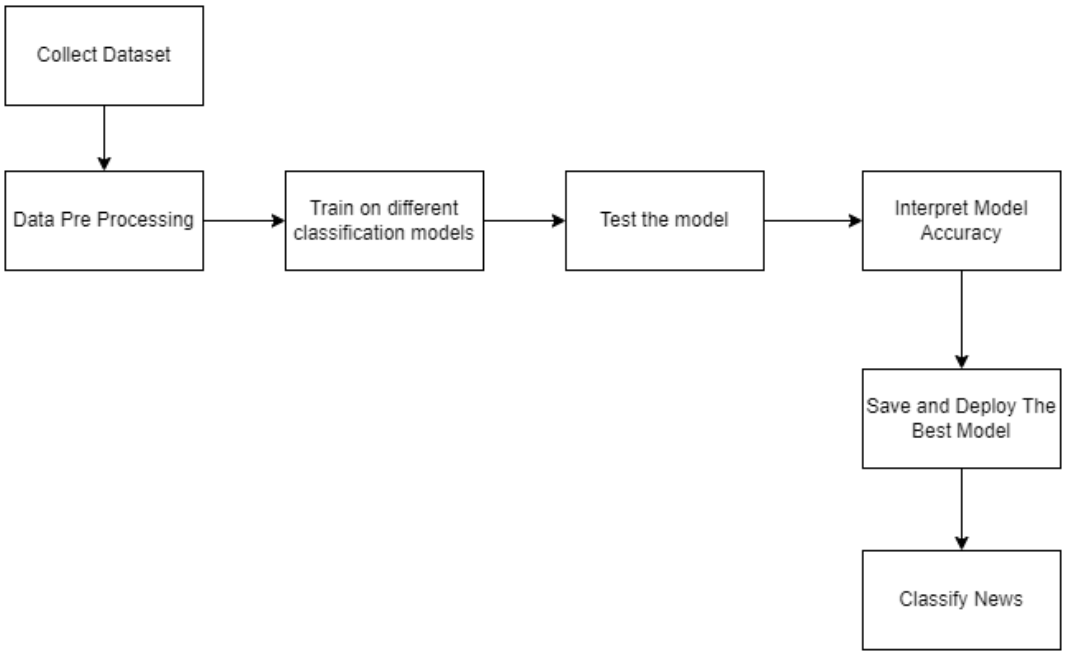


Figure 4.1: High Level Design of the system

The system architecture is as shown in the Figure 4.1. The procedures involved are as:

- Data Collection

  For classification, the dataset was collected from Kaggle for the English news. The dataset was already in a CSV file so it could directly be loaded into a data frame using pandas. The dataset consisted of text and corresponding class. The Nepali news dataset was also collected from Kaggle but unlike the English dataset it was stored as a TXT file under folders of different categories. So, a simple code was

written to read the file and store it in a CSV file in accordance with their respective categories.

For summarization, the dataset was collected from Kaggle again for English and for Nepali it was collected from GitHub. The dataset consisted of news, category and the actual summary.

- Data Pre-Process and Transformation

  For the data preprocessing of the English dataset, First the sentences were tokenized into words. Then the stop words were removed using NLTK.corpus library. Furthermore, the number from the sentences were removed. Finally, Lemmatization was done meaning reducing the words to their root form. E.g.: "Goes"," Went"," Gone" to "Go".

  And for the preprocessing of Nepali dataset, Collection of stop words,numbers and suffixes were made and stored in a text file. This would get loaded later for preprocess. Then a function named word_tokenize was defined that would tokenize a sentence with respect to punctuations: ।', ',', ';', '?', '!', '—', '-', '.'. As the dataset was loaded from text files, Escape characters like new line and whitespace were removed. Then, sentences were tokenized using the default function and the numbers from the sentences in both English and Nepali were removed. Suffix letters were tried to get removed but it didn't seem so fruitful. Hence, it was discarded. After that, the text was vectorized using TFIDF matrix which is Term Frequency-Inverse Document Frequency which is a technique used in information retrieval and text mining to represent text data as a numerical vector.

- Training Models

  Based upon the papers it was found that KNN, MNB and SVM are the most popular models for classification. So, training of the dataset on these models using different combinations of hyperparameters were done. The hyperparameters for KNN were 'N-Neighbors' which are the number of neighbors that will vote for the class of target point and 'Metric' which defines the distance measure to be used. For MNB, the hyperparameters were 'alpha' which is a smoothing parameter that helps to avoid zero probabilities and 'fit_prior' which is a Boolean value that determines whether or not to learn class prior probabilities from the training data. For SVM,

the hyperparameters were ‘C’ which Controls the trade-off between maximizing the margin and minimizing the classification error, ‘Gamma’ which controls the shape of the decision boundary and ‘Kernel’ which specifies the kernel function used to transform the input data into a higher-dimensional space.

- Interpret and Compare Models

  For classification, after training the models, the results of the models were interpreted. The classification model was interpreted on the basis of recall, precision and f1-score. Accuracy is also used. With the help of a confusion matrix visualization was also done and models were also compared against each other.

  For summarization, ROUGE score was used for interpreting which essentially means that it measures how much of the important information in the human-generated summary is captured by the machine-generated summary.

- Save and Deploy the Best Model

  After comparing the models, the best model was selected and it was saved in a pickle file which would be used for deployment of the project.

4.1.1 Refinement of Class Diagram

The refined class diagram is the detailed formed of the class diagram with the addition of precise definition of each attribute and operations of each class.

The Figure 4.2 below shows the refined class diagram of SnapNews. It consists of three classes ScrapeNews, Classify and Summarize. The class of ScrapeNews has a dependency relation with Classify and Summarize class meaning the classes of Summarize () and Classify () are used in the ScrapeNews class in order to classify and then summarize the scraped news.

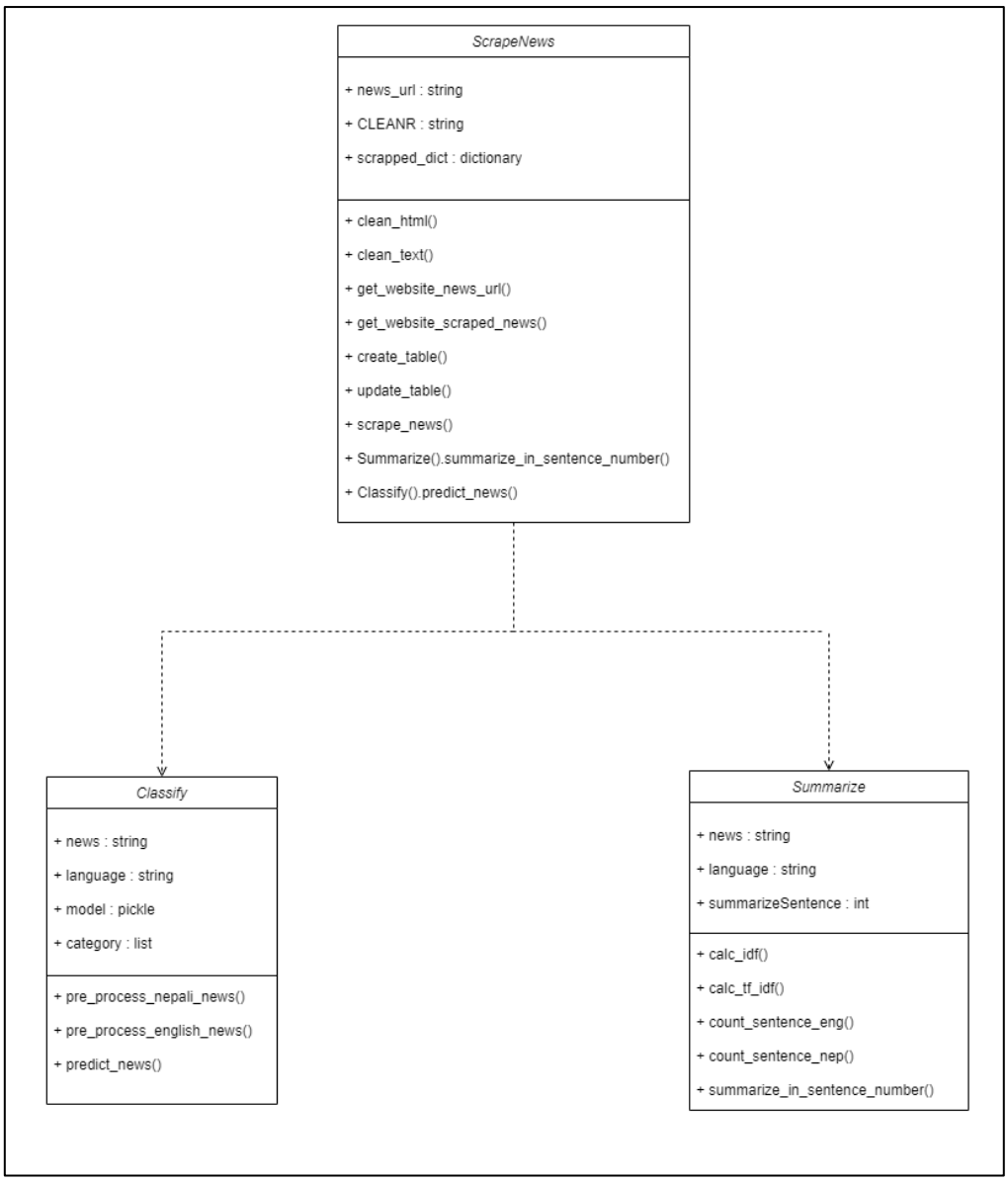


Figure 4.2: Refined Class Diagram

The ScrapeNews has the attributes of news_url which is a string that stores the URL of the news article that is scrapped, CLEANR which is also a string which store the regular expression used for cleaning the HTML content and the scrapped_dict which is a dictionary that stores the scrapped news articles. The methods in this class are all public which cleans the HTML and the text content of the news articles and then create the table for storing the scrapped news along with updating the news.

The Classify class is responsible for classifying the news articles based on the category. It consists of the attributes of a string news which is the news article that needs to be classified, the language attribute is a string attribute which detects the news input to be in

Nepali or English, the model attribute is of type pickle that defines the classification model which classifies the news into either Business, Entertainment, Politics, Tech and Sports. The category attribute is of type list that defines the category of the news article. The methods in this class basically preprocesses the English or Nepali text and then predicts the category of the news article.

The Summarize class summarizes the news articles based on user input. It consists of the attribute's news, language, summarizeSentence where summarizeSentence is of type integer in which a user can put into a value to summarize the content of the news article. The methods in this class calculates the TF-IDF value of the news article sentences and then extracts the important sentences and thus present a summary to the user based upon their inserted summary count.

4.1.2 Refinement of Sequence Diagram

A refined sequence diagram is a type of UML (Unified Modeling Language) diagram that provides a detailed view of the interactions between objects in a system or process. It shows the sequence of messages exchanged between objects or components of a system and the order in which these interactions occur.

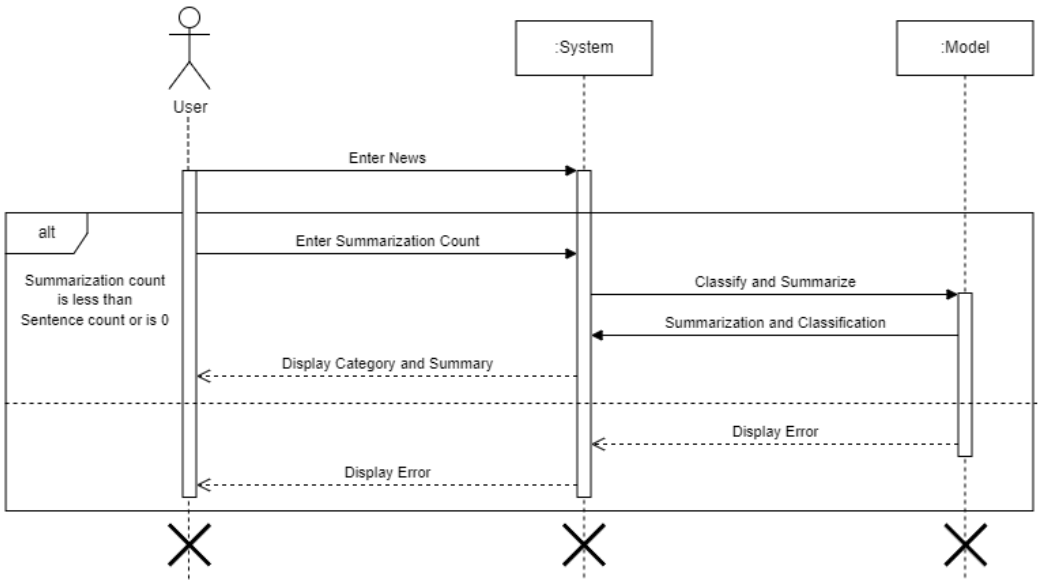


Figure 4.3: Refined Sequence Diagram for Form Input

The above Figure 4.3 is the refined sequence diagram for classifying and summarizing the news articles based upon the form input by the user. The user enters the news they want to classify and summarize along with the summary count they want to input into. After entering the system checks for the summarization count of the user input value with the sentence count of the news article they provided and ensures that the summarization count is not greater than the sentence count of the news article. And then it displays the category by using the classification model and the summary of the news provided.

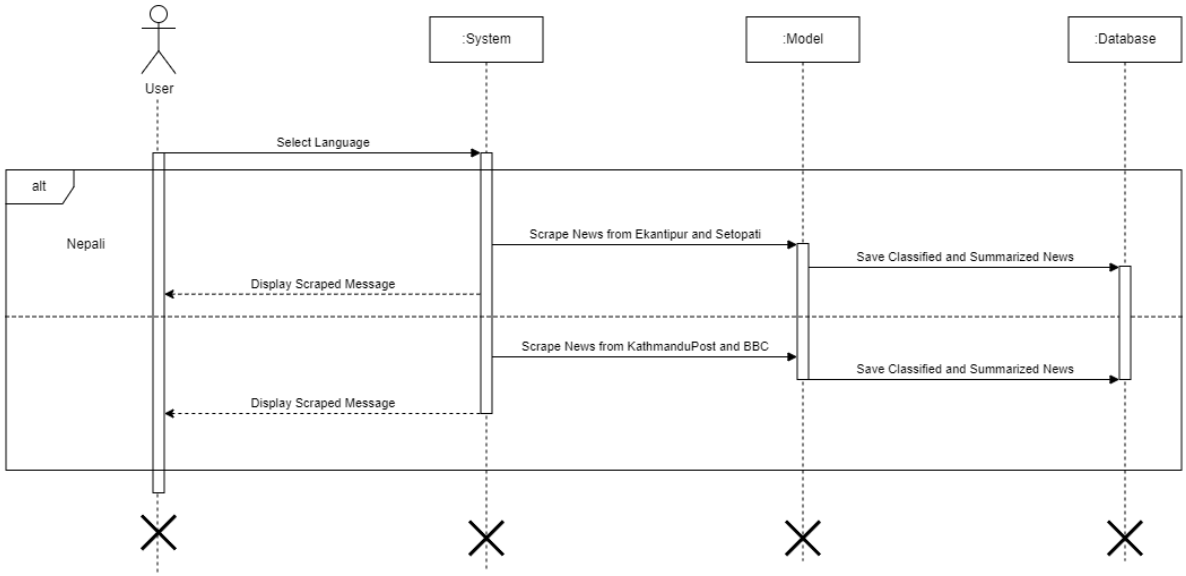


Figure 4.4: Refined Sequence Diagram for Scrape

The above Figure 4.4 is the refined sequence diagram for the scraping. The user first selects the language be it either Nepali or English. If the user selects Nepali as the language, then the system scrapes the news articles from Ekantipur and Setopati as the source and then stores the scrapped news articles into the database by classifying along with summarizing news articles 25% of the total sentence count. Whereas, if the user selects English then the system scrapes the news articles from KathmanduPost and BBC as the source and then stores it into the database by classifying and then summarizing the news articles by 25% of the total sentence count.

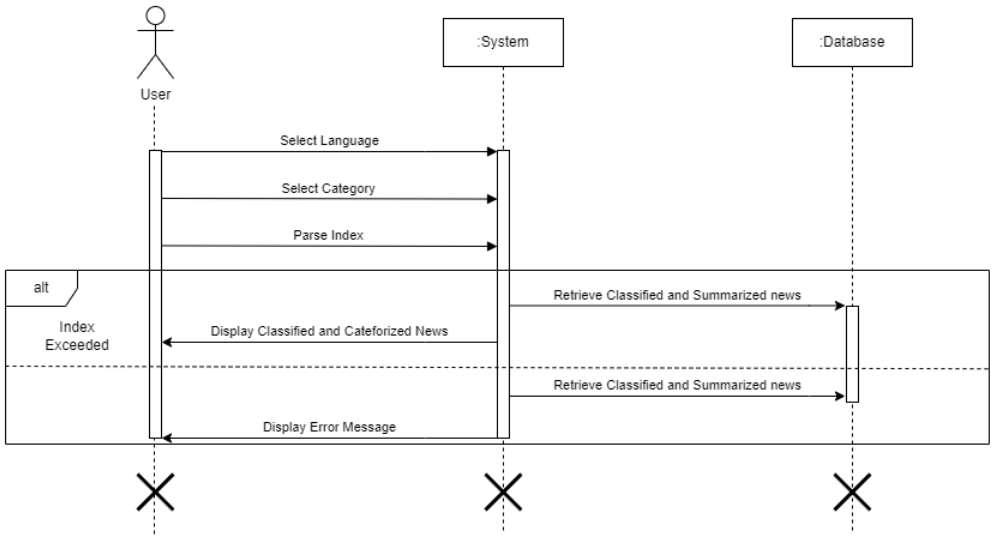


Figure 4.5: Refined Sequence Diagram for View

The above Figure 4.5 is the refined sequence diagram for viewing the scrapped news. The user selects the language they want the news articles to see. The initial category is Business but the user can select any category they want the news to see. The user can then view the news articles by going left and right of the UI and they can also change the category by going up and down of the UI. The index parsed is the news available by scraping. If the index is exceeded then the error is shown to the user else the system retrieves the classified and summarized news and display it to the user.

4.1.3 Refinement of Activity Diagram

A refined activity diagram is a type of activity diagram that provides a more detailed representation of the activities and actions involved in a particular process or workflow. It builds upon the basic activity diagram by providing additional details and steps, allowing for a more thorough understanding of the process.

The Figure 4.6 below is the refined activity diagram for form-based input where an user inputs a news into the form along with the summarization count the system then analyzes the summarization count along with the sentence count of the whole news article and checks whether the user input of the summarization count is greater than that of the sentence count of the news article itself, and also checks if the summarization count is equal to zero if that is the case then the system gives an error to the user else the system uses the classifier

and then summarizer to classify the category of the news article and then summarize it based upon the users input. Then it displays the summarized and the classified news to the user.

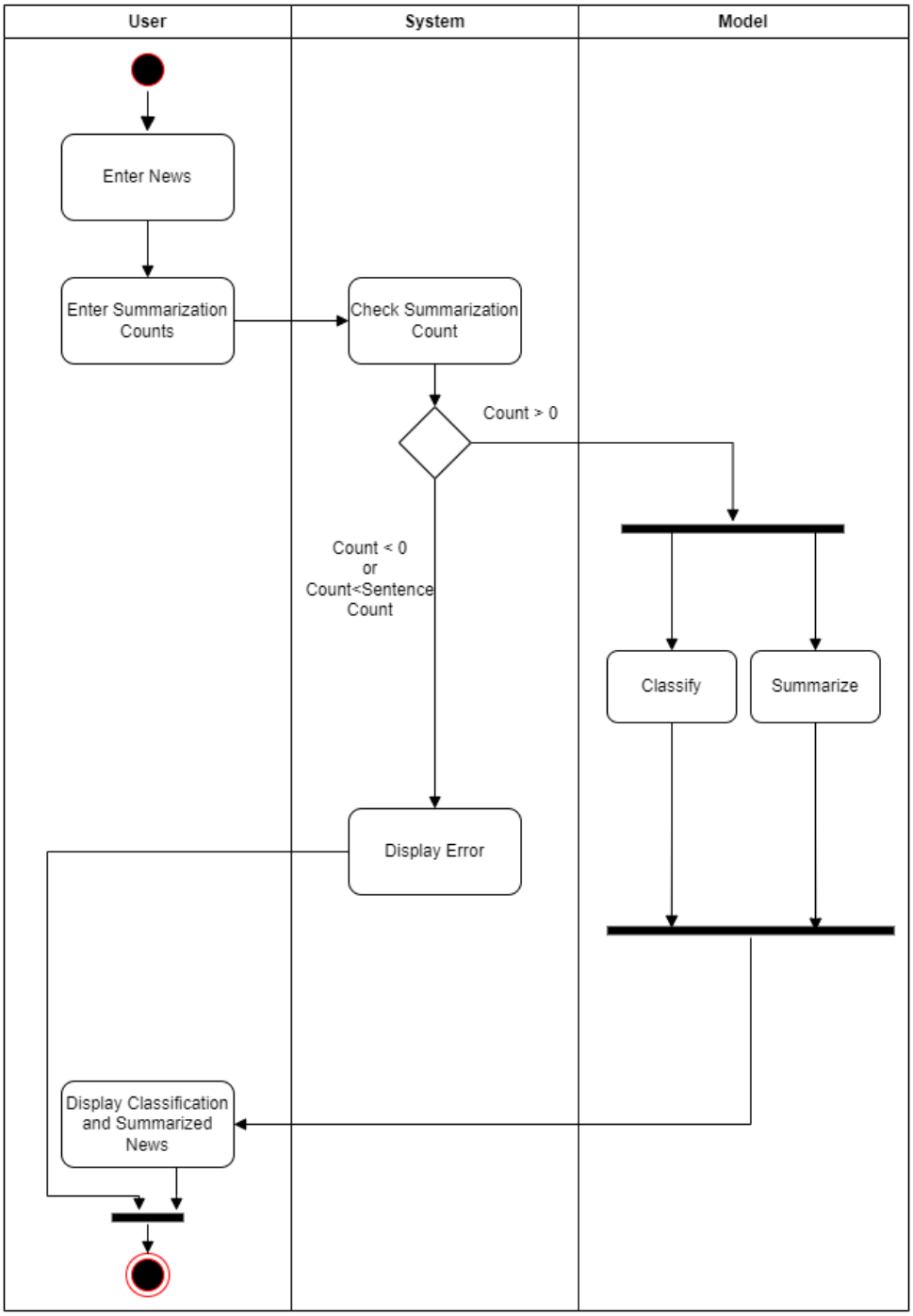


Figure 4.6: Refined Activity Diagram for form input

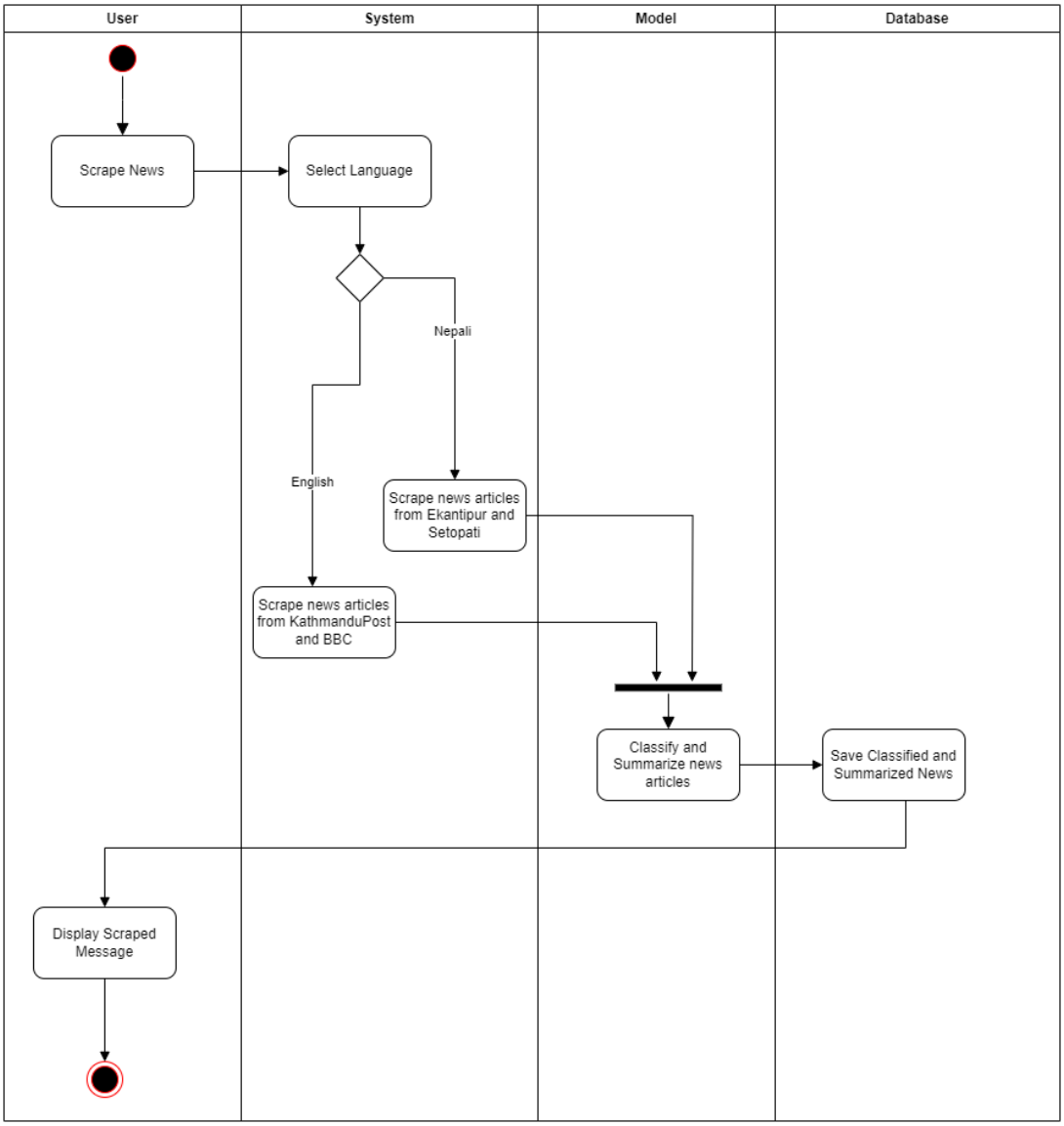


Figure 4.7: Refined Activity Diagram for scraping

The above Figure 4.7 is the refined activity diagram for scraping of the news. The user first initiates the scrape news process where the user can select the language of the news article to be scrapped. The user then chooses between English and Nepali language. If the user picks the Nepali language, then the news articles from the Ekantipur and Setopati would be scraped by the scraper module. And, if the user selects the English language, then the news articles from the Kathmandu Post and the BBC would get scraped. After scraping the news, the news articles would be classified and summarized through the classify and summarize modules and then the classified and the summarized news articles would be saved in the database which would then be displayed to the user.

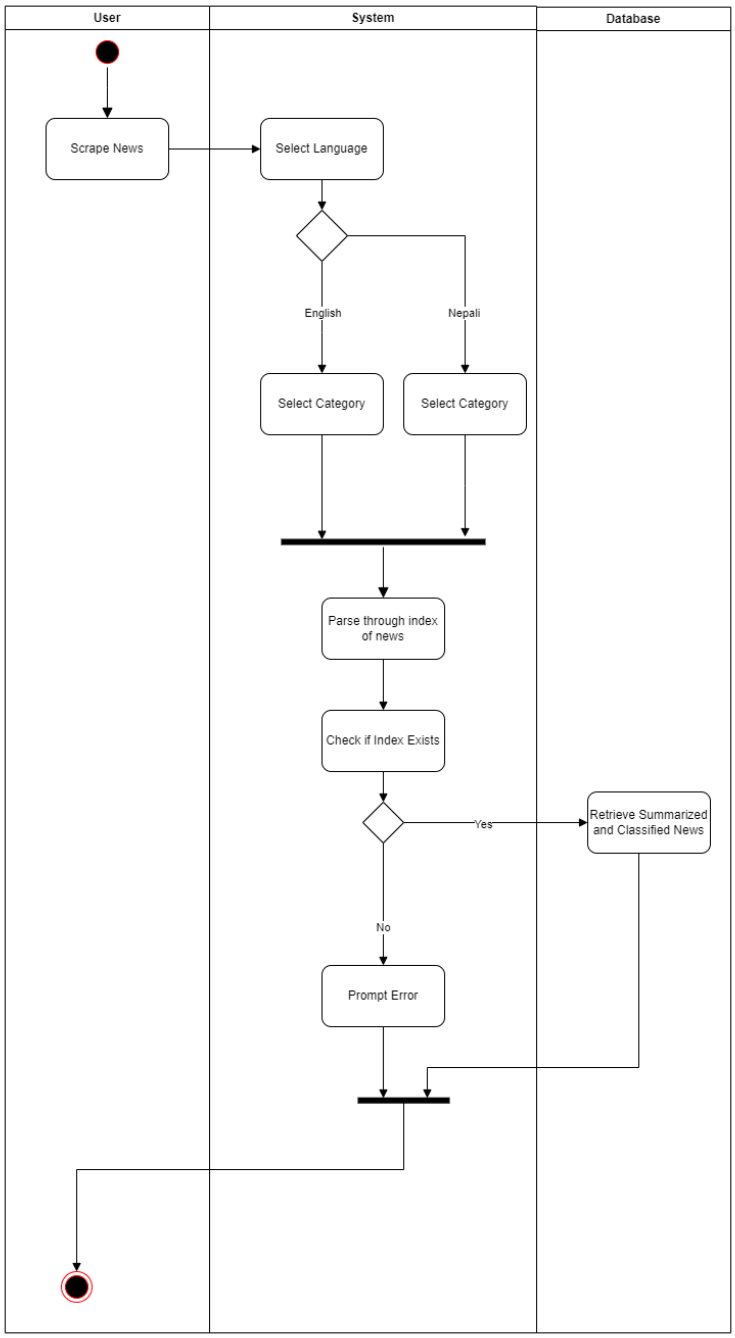


Figure 4.8: Refined Activity Diagram for view

The above Figure 4.8 is the refined activity diagram for the view. The user can first select between the two languages: English and Nepali. Whichever language the user selects the scraped news along with its summary is shown to the user. The title of the news, the date, the link to the news would be shown to the user. The user can view through the five categories of the news articles along with the news articles of the same category. If the

index of the news articles exceeds then an error is shown to the user else the user can keep viewing the news articles.

4.1.4 Component Diagram

A component diagram is a type of UML diagram that shows the structural relationships and dependencies between the components of a software system. It illustrates how software components are connected and interact with each other within a system. Components can represent individual modules, libraries, executables, or other parts of a system.

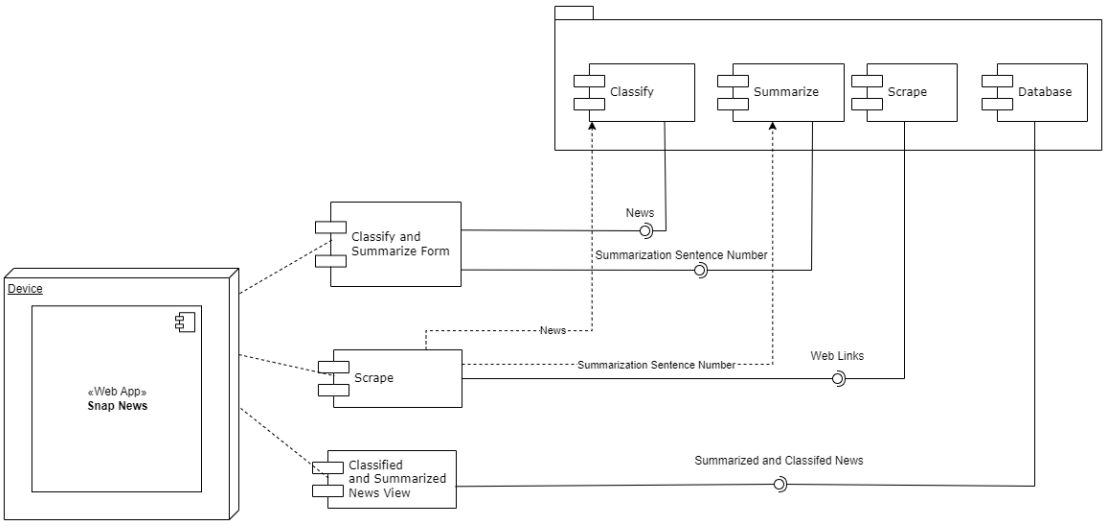


Figure 4.9: Component Diagram

The Figure 4.9 above is the component diagram which shows how the components are connected to each other and how they are dependent. The system consists of three components namely: Classify and Summarize Form, Scrape and Classified and Summarized News Views. The classify and summarize form has interface with the classify module through news and through summarization sentence with summarize. The scrape component has dependency with Classify and summarize and interfaces with Scrape module through web links. Finally, The classified and summarized news view interfaces with database module through the summarized and classified news.

4.1.5 Deployment Diagram

Deployment diagram are UML structural diagrams that shows the relationships between the hardware and software components in the system and the physical distribution of the processing i.e., Deployment diagram are used to visualize the topology of the physical components of the system where software components are deployed.

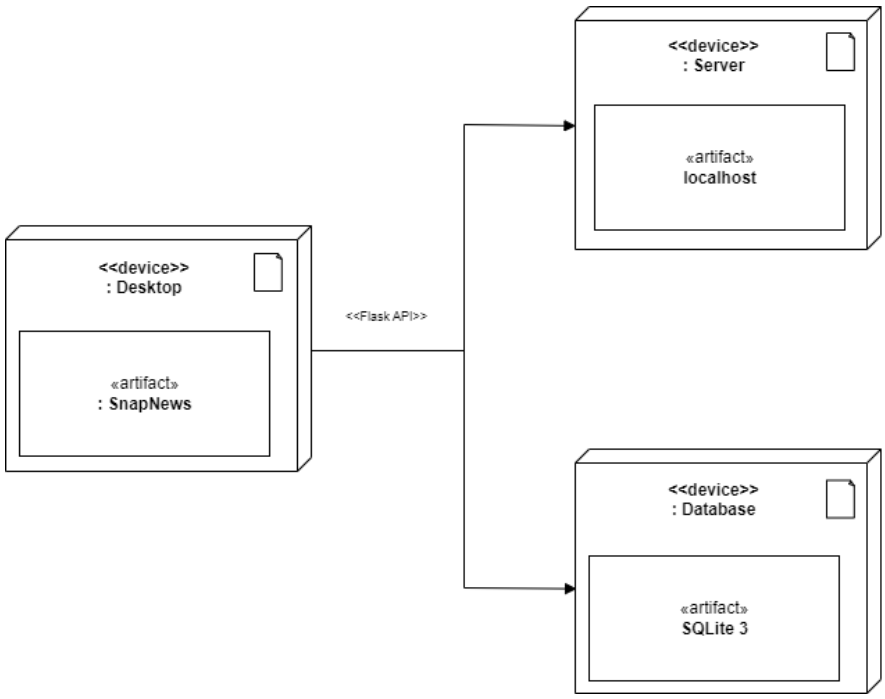


Figure 4.10: Deployment Diagram

The above Figure 4.10 shows the deployment diagram of SnapNews. The server node hosts the SnapNews application and the client node is where the user interacts with the application. The database node is where the SnapNews system stores its data. The client side and the server side are connected through Flask API for communicating.

4.2 Algorithm Description

The project implements the following algorithms:

- **KNN for Classification**

K-Nearest Neighbors (KNN) is a classification and regression machine learning algorithm. It is a non-parametric method that works by locating the k closest data points in the training

dataset to a given data point in the test dataset and then using the majority class or average value of those k-nearest neighbors as the predicted class or value for the test data point. The hyperparameters for KNN were 'N-Neighbors' which are the number of neighbors that will vote for the class of target point and 'Metric' which defines the distance measure to be used.

The steps involved for KNN are as follows:

1. Data Preparation: First, we need a labeled dataset consisting of feature vectors and their corresponding class labels. The features represent the characteristics or attributes of the data points, while the class labels indicate the category or class to which each data point belongs.
2. Choosing the Value of K: The parameter "k" represents the number of nearest neighbors to consider for classification. It is important to select a suitable value for K, as it affects the algorithm's performance. A larger value of K provides smoother decision boundaries but may lead to misclassifications, while a smaller value of K makes the decision boundary more sensitive to outliers.
3. Distance Metric: To determine the nearest neighbors, we need to define a distance metric to measure the similarity between data points. The most commonly used distance metric is Euclidean distance, given by the formula:
   Euclidean Distance

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

   Manhattan Distance

$$d(A,B) = \sum_{i=1}^{n}|A_i - B_i|$$

   Cosine Similarity

$$\cos\theta = \frac{A.B}{\|A\|\|B\|}$$

   Here, A and B represent two data points, and n is the number of features.

4. Finding Nearest Neighbors: For a given test data point, the algorithm calculates the distance to all the other data points in the training set using the chosen distance metric. It then selects the k nearest neighbors based on the calculated distances.
5. Majority Voting: Once the k nearest neighbors are identified, the algorithm determines the majority class among them. Each neighbor's class label is given equal weight, and the class with the highest count is assigned to the test data point.
6. Classification: Finally, the algorithm assigns the predicted class label to the test data point based on the majority voting result.

The KNN algorithm is simple yet effective, offering a straightforward way to classify new data points based on their proximity to existing labeled data.

- **MNB for Classification**

Multinomial Naive Bayes is a probabilistic machine learning algorithm used for text classification and other tasks involving discrete data. It is founded on the Bayes theorem and the assumption of conditional independence between the features of the given class. For MNB, the hyperparameters were 'alpha' which is a smoothing parameter that helps to avoid zero probabilities and 'fit_prior' which is a Boolean value that determines whether or not to learn class prior probabilities from the training data.

The steps involved in MNB classification are as follows:

1. Data Preparation: Like any classification algorithm, MNB requires a labeled dataset with feature vectors and class labels.
2. Probability Calculation: MNB calculates the probability of a document or data point belonging to each class by applying Bayes' theorem. The formula for calculating the probability of a class given the features is:
   MNB Formula

$$P(C_k|x) = \frac{P(C_k) \times P(x|C_k)}{P(x)}$$

   Here, $P(C_k|x)$represents the probability of the class C_k given the features x, $P(C_k)$is the prior probability of class C_k, $P(x|C_k)$ is the likelihood of features x given class C_k, and P(x) is the probability of features x.
3. Independence Assumption: MNB assumes independence between features. It calculates the likelihood of a feature given a class using the formula:

MNB Likelihood Formula

$$P(x|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

Here, $P(x_i|C_k)$ represents the probability of the i[th] feature given class C_k.

4. Decision Rule: To classify a new data point, MNB uses the maximum a posteriori (MAP) decision rule, which selects the class with the highest posterior probability. It assigns the data point to the class $C_k$with the maximum P(C_k|x).

- **SVM for Classification**

SVM stands for Support Vector Machine, which is a popular type of machine learning algorithm used for classification and regression analysis. In SVM, the algorithm tries to find the best hyperplane (a decision boundary) that separates the data points into different classes. The hyperplane is selected such that the distance between the hyperplane and the nearest data points from each class (called support vectors) is maximized. This distance is known as the margin, and SVM aims to maximize this margin.

The steps for implementing SVM are as follows:

1. Data Preparation: Similar to other classification algorithms, SVM requires a labeled dataset with feature vectors and class labels.
2. Hyperplane Construction: SVM finds an optimal hyperplane that maximally separates the data points of different classes. The hyperplane is defined by the formula:
   SVM Hyperplane Formula

$$w \cdot x - b = 0$$

   Here, w represents the weight vector perpendicular to the hyperplane, x is the feature vector of a data point, and b is the bias or intercept term.
3. Margin Maximization: SVM aims to maximize the margin between the hyperplane and the nearest data points of different classes. The margin is given by the formula:
   SVM Margin Formula

$$\max\left(\frac{2}{\|w\|}\right)$$

SVM finds the hyperplane that maximizes this margin.

4. Support Vectors: The data points that lie closest to the hyperplane and determine its position are called support vectors. They are used to calculate the margin and are crucial for classification.
5. Kernel Trick: SVM can efficiently handle non-linearly separable data by applying the kernel trick. The kernel function computes the inner product between the feature vectors in a high-dimensional space without explicitly transforming the data. Popular kernel functions include linear, polynomial, and radial basis function (RBF) kernels.
6. Classification: Once the hyperplane is constructed, SVM classifies new data points based on which side of the hyperplane they fall on. If a data point lies on one side, it is assigned to one class; otherwise, it is assigned to the other class.

It's important to note that SVM allows for soft-margin classification by introducing a penalty term for misclassified points, as well as C parameter for controlling the trade-off between margin maximization and training errors.

- **Extractive Summary using TFIDF Matrix**

TFIDF stands for Term Frequency-Inverse Document Frequency. It is a technique used in information retrieval and text mining to represent text data as a numerical vector. The goal is to transform unstructured text data into structured numerical data that can be processed by machine learning algorithms. A TFIDF matrix is a matrix that represents the importance of each word in a document corpus. The matrix is constructed by calculating the TFIDF score for each word in each document. The TFIDF score is calculated as follows:

TF (Term Frequency): the number of times a word appears in a document divided by the total number of words in the document.

IDF (Inverse Document Frequency): the logarithm of the total number of documents in the corpus divided by the number of documents that contain the word.

The TFIDF score for a word in a document is the product of the TF and IDF scores. This score represents how important the word is in the document and how rare the word is in the corpus. The TFIDF matrix is typically a sparse matrix since most words do not appear in most documents. The rows of the matrix represent the documents in the corpus, and the columns represent the words. Each entry in the matrix represents the TFIDF score for a particular word in a particular document.

The TFIDF matrix can be used as input to machine learning algorithms for tasks such as text classification, clustering, and information retrieval. It can also be used for data exploration and visualization.

Let there be 'n' number of document in a document corpus, 't' be a specific word and d_0, d_1, ...d_n be each document in the document corpus.

1. First, the IDF value of each word was calculated. Here the IDF is first calculated by calculating the df(t). df(t) is number of times a specific word has been occurred in different documents. Then IDF is calculated as

$$idf(t) = log_e\left(\frac{1+n}{1+df(t)}\right) + 1$$

   Thus, The IDF values of all t's occurred in our document corpus was calculated.

2. Now, TFIDF value for each document was calculated and it is calculated as

$$tfidf(t) = tf(t) * idf(t)$$

   Here $tf(t)$ is number of times a specific word 't' has occurred in that sentence

3. Now, the documents were sorted in ascending order in values of TFIDF and the most important documents have the highest TFIDF values and was used as summarized sentences.

CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation Overview

Process model used

Incremental delivery is a process model that involves breaking down a project into smaller, more manageable components and delivering these components incrementally over time. Each increment builds upon the previous one, adding new features and functionality until the final product is complete.

As an Agile methodology, incremental delivery emphasizes collaboration between developers, stakeholders, and customers. The development team works closely with the customer to identify their needs and requirements and deliver working software quickly and frequently. This allows the customer to see progress and provide feedback along the way, which can be incorporated into the next iteration.

5.1.1 Tools Used

5.1.1.1 Front End Tools

The front-end of the Snap News project was built using ReactJS, which is a popular JavaScript library for building user interfaces. ReactJS is known for its ability to create reusable UI components and its focus on performance, making it a great choice for building complex web applications. With addition to ReactJS, HTML/CSS was used for the basic styling and structure of the webpage. Furthermore, Axios was used to make HTTP requests from the frontend to the backend. Also, React-Router was used in order for navigating between the pages.

5.1.1.2 Back End Tools

The back-end of the Snap News project was built using Flask, which is a popular Python web framework for building APIs and web applications. Flask is known for its simplicity and ease of use, making it a great choice for building small to medium-sized web

applications. With addition to Flask, NTLK was used for training the dataset for preprocessing, Scikit Learn for training the machine learning model, Pandas and Numpy to load and store the data and to read the datasets and BeautifulSoup for scraping the news articles.

5.1.2 Modules Description

This project implements following modules:

- **Classification Module**

This module is used to classify the news articles into its respective category business, entertainment, politics, sports and tech. Since SVM produced the highest accuracy, it was used for classification for scraping. It classifies both English and Nepali news. It takes news as input. The classification module has function pre_process_nepali_news and pre_process_english_news. This function respectively pre_processes input news. The function pedict_news then classifies the processed news to its respective class.

- **KNN For Classification**

```
def predict(self, X_test):
    neighbors = []
    for x in X_test:
        if self.metric == 'euclidean':
            distances = np.sqrt(np.sum((x - self.X_train)**2, axis=1))
        elif self.metric == 'manhattan':
            distances = np.sum(np.abs(x - self.X_train), axis=1)
        elif self.metric == 'cosine':
            distances = [cosine(x, data) for data in self.X_train]

        y_sorted = [y for _, y in sorted(zip(distances, self.y_train), key=lambda pair: pair[0])]
        neighbors.append(y_sorted[:self.n_neighbors])

    return list(map(most_common, neighbors))

def score(self, X_test, y_test):
    y_pred = self.predict(X_test)
    accuracy = sum(y_pred == y_test) / len(y_test)
    return accuracy
```

Figure 5.1: Using KNN to Classify

The predict method is used to make predictions on the test data (X_test). It initializes an empty list called neighbors to store the predicted labels for each data point in X_test. It iterates over each data point x in X_test. Depending on the specified distance metric (self.metric), it calculates the distances between x and all the data points in the training set

(self.X_train). Depending upon the metric it calculates the distance between x and all data points in the training set. The distances are sorted in ascending order, and the corresponding labels (self.y_train) are sorted accordingly using sorted(zip(distances, self.y_train), key=lambda pair: pair[0]). The most_common function is applied to the self.n_neighbors nearest labels to determine the most frequently occurring label. The predicted labels for each data point in X_test is appended to the neighbors list. Finally, the method returns the neighbors list, which contains the predicted labels for each data point in X_test.

The score method is used to evaluate the accuracy of the classifier's predictions. It calls the predict method with the test data (X_test) to obtain the predicted labels (y_pred). It compares the predicted labels (y_pred) with the true labels (y_test) and calculates the accuracy by dividing the number of correct predictions by the total number of predictions. The calculated accuracy is returned as the score of the classifier.

- **MNB For Classification**

```
def fit(self, X, y):
    self.classes = list(set(y))
    self.n_classes = len(self.classes)
    self.n_features = X.shape[1]
    self.feature_counts = {c: np.ones(self.n_features) * self.alpha for c in self.classes}
    self.class_counts = {c: 0 for c in self.classes}
    self.prior_probs = {c: 0 for c in self.classes}

    for i in range(len(y)):
        c = y[i]
        self.class_counts[c] += 1
        self.feature_counts[c] += X[i]

    if self.fit_prior:
        n_samples = len(y)
        for c in self.classes:
            self.prior_probs[c] = self.class_counts[c] / n_samples

    self.feature_probs = {c: np.log(self.feature_counts[c] / sum(self.feature_counts[c])) for c in self.classes}
```

Figure 5.2: Fitting in MNB

In the fit method, it initializes important attributes of the classifier, such as classes (unique classes in the labels y), n_classes (number of classes), n_features (number of features in the input data X), feature_counts (dictionary to store feature counts for each class with Laplace smoothing), class_counts (dictionary to store the count of each class), and prior_probs (dictionary to store the prior probability of each class). It iterates over the training samples and updates the class and feature counts. It increments the class count and adds the feature values to the corresponding feature count for that class.

If fit_prior is set to True, it calculates the prior probabilities for each class by dividing the class count by the total number of samples. It computes the feature probabilities for each class by taking the logarithm of the smoothed feature counts divided by the sum of feature counts for that class.

```
def predict(self, X):
    n_samples = X.shape[0]
    y_pred = []
    for i in range(n_samples):
        scores = {c: np.sum(self.feature_probs[c] * X[i]) + np.log(self.prior_probs[c]) for c in self.classes}
        y_pred.append(max(scores, key=scores.get))
    return y_pred

def score(self, X, y):
    y_pred = self.predict(X)
    return sum(y_pred[i] == y[i] for i in range(len(y))) / len(y)
```

Figure 5.3: Predict and Score in MNB

In the predict method, it takes unseen data points X as input and returns the predicted labels. It iterates over each data point and computes the log scores for each class. The scores are calculated by summing the product of feature probabilities and the corresponding feature values in the data point, and adding the logarithm of the prior probability. It selects the class with the maximum score as the predicted label and appends it to the y_pred list.

In the score method, it takes the test data X and true labels y as input and returns the accuracy of the classifier's predictions. It calls the predict method to obtain the predicted labels. It compares the predicted labels with the true labels and calculates the ratio of correct predictions to the total number of samples.

- **SVM For Classification**

```
def fit(self, X, y):
    self.X = X.copy()
    self.y = y * 2 - 1
    self.lambdas = np.zeros_like(self.y, dtype=float)
    self.K = self.kernel(self.X, self.X) * self.y[:,np.newaxis] * self.y

    for epoch in range(self.max_iter):
        # print(epoch,end="\r")
        for idxM in range(len(self.lambdas)):
            idxL = np.random.randint(0, len(self.lambdas))
            Q = self.K[[[idxM, idxM], [idxL, idxL]], [[idxM, idxL], [idxM, idxL]]]
            v0 = self.lambdas[[idxM, idxL]]
            k0 = 1 - np.sum(self.lambdas * self.K[[idxM, idxL]], axis=1)
            u = np.array([-self.y[idxL], self.y[idxM]])
            t_max = np.dot(k0, u) / (np.dot(np.dot(Q, u), u) + 1E-15)
            self.lambdas[[idxM, idxL]] = v0 + u * self.restrict_to_square(t_max, v0, u)

    idx, = np.nonzero(self.lambdas > 1E-15)
    self.b = np.mean((1.0 - np.sum(self.K[idx] * self.lambdas, axis=1)) * self.y[idx])
```

Figure 5.4: Fit Method in SVM

The method takes as input the training data X (input features) and y (corresponding labels). The X data is copied to ensure the original data remains unchanged. The labels y is transformed to -1 and 1 to represent different classes. Some memory variables are initialized, including lambdas to store the importance of each training example and K to store the pairwise kernel values between the training examples. The training process iterates for a specified number of epochs, which controls the number of iterations the algorithm performs on the data. Within each epoch, a random pair of training examples is selected. The kernel function is applied to compute the similarity or distance measure between the selected examples. The algorithm calculates the difference between the current prediction and the actual label for the selected examples. Using optimization techniques, the algorithm updates the importance values (lambdas) based on the difference and the kernel values. After all the iterations, the algorithm identifies the support vectors by finding the examples with non-zero importance values. The bias term **b** is computed based on the support vectors and their importance values.

```
def decision_function(self, X):
    return np.sum(self.kernel(X, self.X) * self.y * self.lambdas, axis=1) + self.b
```

Figure 5.5: Decision Function in SVM

This method calculates the decision values or scores for each sample in the input X. It measures the distance of each sample from the decision boundary of the SVM model. The decision values are computed by taking the dot product of the kernel values between the input samples X and the training samples self.X, and multiplying them with the corresponding importance values self.y and self.lambdas. The decision values are then adjusted by adding the bias term self.b. The resulting decision values represent the signed distance of each sample from the decision boundary, with positive values indicating one class and negative values indicating the other.

```
def predict(self, X):
    return (np.sign(self.decision_function(X)) + 1) // 2

def score(self, X_test, y_test):
    y_pred = self.predict(X_test)
    accuracy = sum(y_pred == y_test) / len(y_test)
    return accuracy
```

Figure 5.6: Predict and Score in SVM

This predict method uses the decision_function to make predictions for the input samples X. It applies a sign function to the decision values obtained from decision_function to determine the predicted class label for each sample. The sign function converts positive values to 1 and negative values to -1. Since the SVM model used in this code is a binary classifier, the output of the sign function is then adjusted by adding 1 and dividing by 2 to obtain the final predicted class labels. The resulting labels are either 0 or 1, representing the predicted class for each input sample.

This score method evaluates the accuracy of the predictions made by the predict method. It compares the predicted class labels for the test samples X_test with the true class labels y_test. The method calculates the accuracy by dividing the number of correctly predicted samples by the total number of samples in y_test. The accuracy score reflects the performance of the SVM model in correctly classifying the test samples.

The Figure 5.7 below is a snapshot of an English news of type Politics. The output from different model is given as:

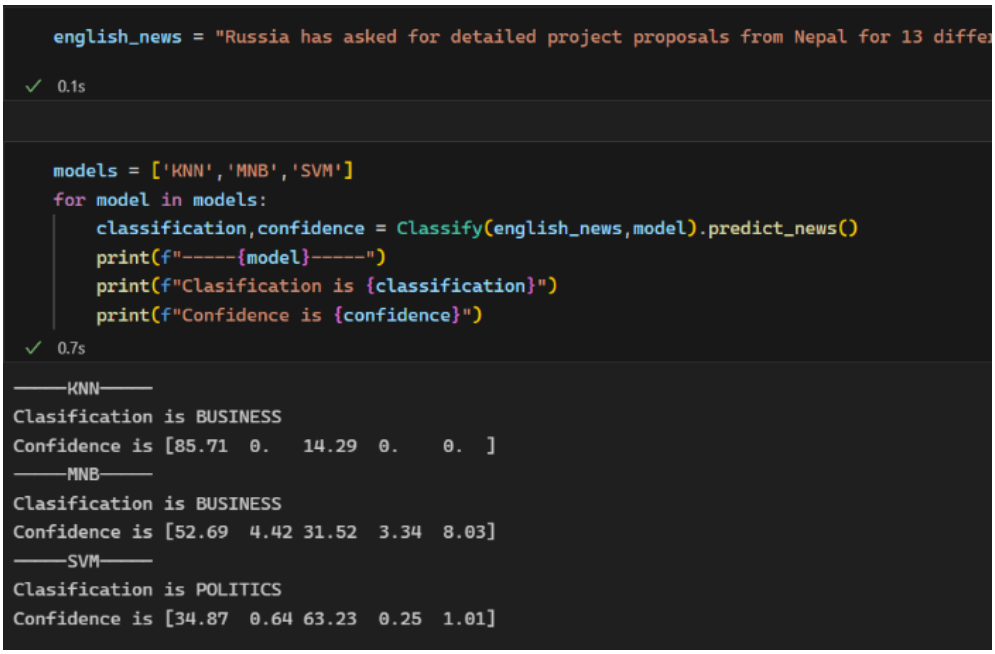


Figure 5.7: Classification for English News

Here only SVM correctly classifies the news. While this is only an example of testing the model. Later the models can be evaluated.

Next The Figure 5.8 below is a snapshot of a Nepali news of type Sport. The output from different model is given as:

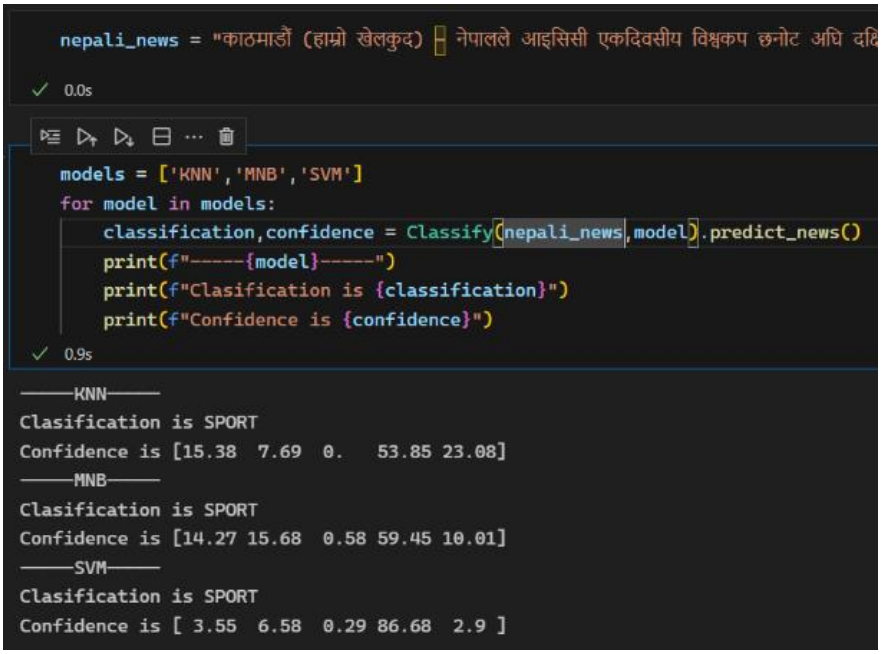


Figure 5.8: Classification for Nepali News

Here all model classifies news as Sports. Further evaluation of different model is to be done.

- **Summarization Module**

This module is used to extract important words from the given text and thus provide the user with an extractive summary. It also takes news as input. It consists of functions calc_idf, calc_tf_idf, count_sentence_eng, count_sentence_nep and summarize_in_sentence_number. The function calc_idf calculates the IDF value for a word in that document corpus and calc_tf-idf calculates the TF-IDF value for the word in that sentence. The functions count_sentence_nep and count_sentence_eng counts the number of sentences in each language. Finally, the summarize_in_sentence_number takes number as input to which the news is to be classified.

The summarization for English news of count 14 is summarized to 3 sentences. Sentences 2, 12 and 13 were chosen. Their tf-idf values were 5.91, 5.58 and 5.06.

Thus, they were selected as summarization.

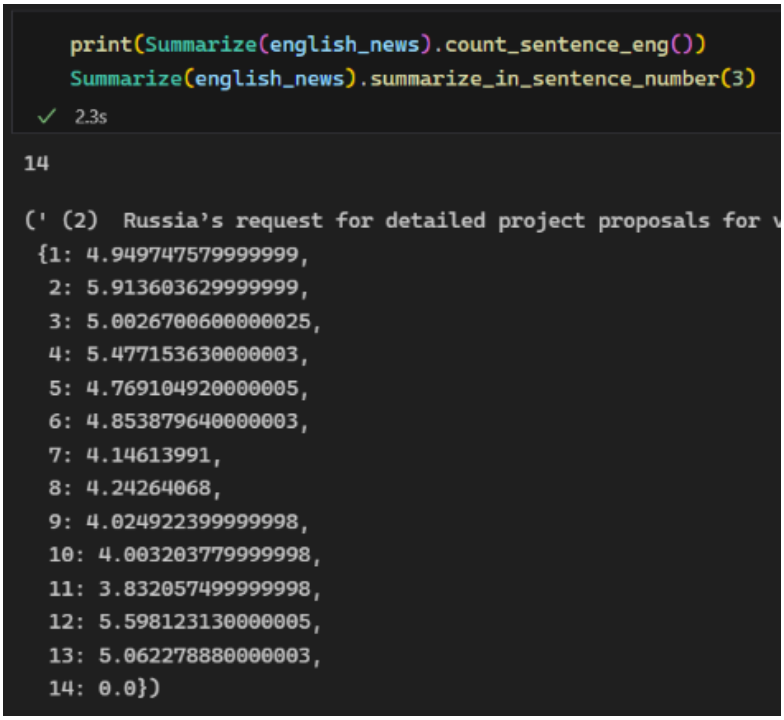


Figure 5.9: English Summarization with TF-IDF Values

The summarization for Nepali news of count 23 is summarized to 3 sentences. Sentences 1, 2 and 6 were chosen. Their tf-idf values were 4.43, 4.09 and 3.8. Thus, they were selected as summarization.

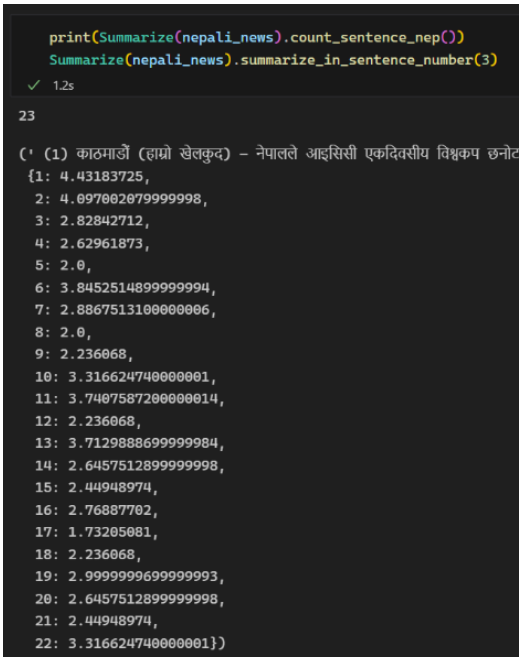


Figure 5.10: Nepali Summarization with TF-IDF Values

- **Scrape Module**

sThis module is used to scrape the news articles off of the different news site and provide a central location for the user to view the different news. Scrape Module is different for scraping EKantipur, Setopati, BBC and The Kathmandu Post. Each module consists of function cleanhtml, clean_text, get_name_news_url, get_name_scraped_news, create table and update_table. Function cleanhtml cleans the scraped news of any html tags. The function clean_text further processes the text. The get_name_news_url retrieves all possible urls and stores them in a list. Now from the stored list get_name_scraped_news scrapes all news articles. Finally create_table creates a new table english_news or nepali_news if it doesnot exists and update_table inserts new classified and summarized news to english_news or nepali_news table. The classification is done with seventy percent threshold. If a news classified has confidentiality of seventy percent or more the news will be categorized on so category else the category is set as others.

5.2 Testing

Testing is the process of evaluating and verifying whether the developed software or application works properly or not i.e., whether there is match between the actual results and expected results or not. Testing is carried out during the development of the software.

5.2.1 Unit Testing

Unit testing is the part of the testing methodology which includes testing of individual software modules as well as the components that make up the entire software. The purpose is to validate each unit of the software code so that it performs as expected.

Table 5.1: Test Cases for Input Validation

S. N	Test Cases	Test Description	Input Test Data	Expected Results	Actual Result	Remarks
1.	TC-IV-1	Handle different types of inputs, such as text data of	English News	Classified English News	Classified English News	Pass

		varying length and format.				
2.	TC-IV-2	Handle different types of inputs, such as text data of varying length and format.	Nepali News	Classified Nepali News	Classified Nepali News	Pass

The Table 5.1 above shows the test cases for input validation. At first the English news of varying length was input into the form which gave the category of the news. Furthermore, the same was done for the Nepali news while it took a bit of time for Nepali news, it did give the category of the news text inputted. The Figures 5.11 and 5.12 show the category of the news thus resulted from different models.

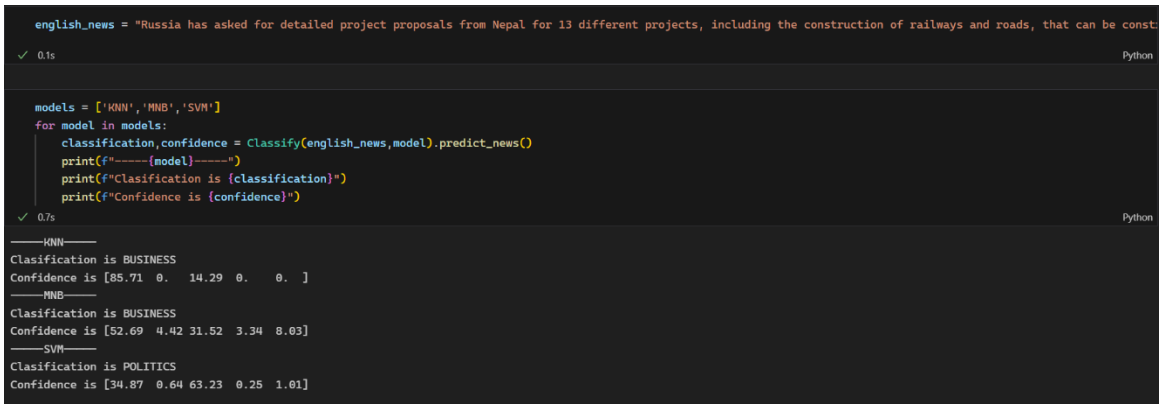


Figure 5.11: Classification Validation for English News

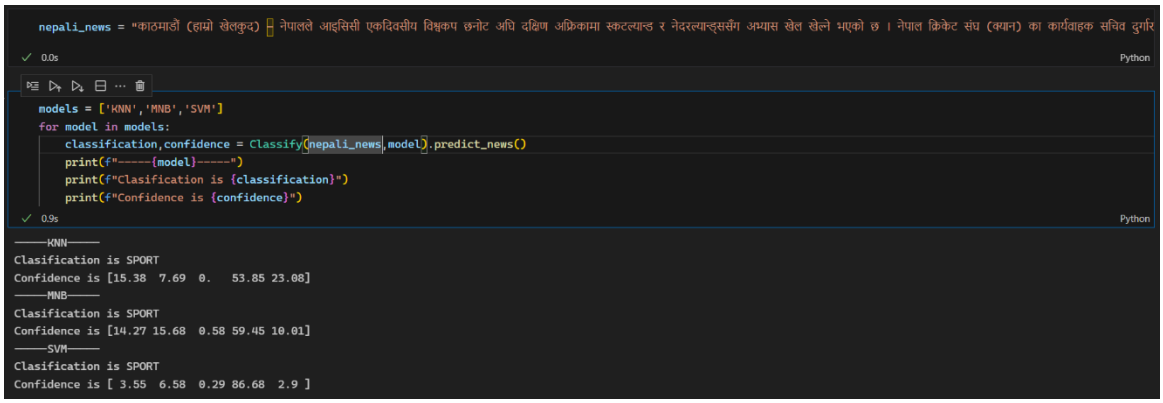


Figure 5.12: Classification Validation for Nepali News

Table 5.2: Test Cases for Model Accuracy

S. N	Test Cases	Test Description	Input Test Data	Expected Results	Actual Result	Remarks
1.	TC-MA-1	Tested on Test Data of our dataset	English Test Data	Good Accuracy	98% Accuracy	Pass
2.	TC-MA-1	Tested on Test Data of 908 dataset	Nepali Test Data	Good Accuracy	95% Accuracy	Pass

The Table 5.2 above shows the test cases done for the model accuracy. Datasets of both languages were tested. Using Recall, Precision and f1-score the accuracy of model was tested. Greater accuracy was expected. For English, accuracy was 98% whereas for Nepali dataset it was 95%.

```
Train Accuracy Score : 100
Test Accuracy Score  : 98

              precision    recall  f1-score   support

     business       1.00      0.98      0.99        65
entertainment       0.97      0.98      0.97        58
     politics       0.95      0.98      0.97        59
        sport       1.00      1.00      1.00        61
         tech       1.00      0.96      0.98        55

     accuracy                           0.98       298
    macro avg       0.98      0.98      0.98       298
 weighted avg       0.98      0.98      0.98       298
```

Figure 5.13: Accuracy for English News using SVM Model

```
Train Accuracy Score : 100
Test Accuracy Score  : 95

              precision    recall  f1-score   support

     business       0.92      0.92      0.92       202
entertainment       0.97      0.94      0.95       224
     politics       0.95      0.98      0.96        97
        sport       0.96      0.99      0.98       179
         tech       0.94      0.93      0.93       206

     accuracy                           0.95       908
    macro avg       0.95      0.95      0.95       908
 weighted avg       0.95      0.95      0.95       908
```

Figure 5.14: Accuracy for Nepali News using SVM Model

Table 5.3: Test Cases for Performance

S. N	Test Cases	Test Description	Input Test Data	Expected Results	Actual Result	Remarks
1.	TC-P-1	System performs efficiently under normal and peak loads.	English News Article of varying lengths.	Classifies and summarizes the news under minimal time.	Classified the news under 30 secs.	Pass
2.	TC-P-2	System performs efficiently under normal and peak loads.	Nepali News Article of varying lengths.	Classifies and summarizes the news under minimal time.	Classified the news under 30 secs.	Pass

The Table 5.3 above shows the test cases for the performance of the system. Varying length of the news was inputted of both the languages. While the classification didn't take much time, a bit more time was taken for summarization process. But as a whole around 30 secs time was taken accounting for both classification and summarization.

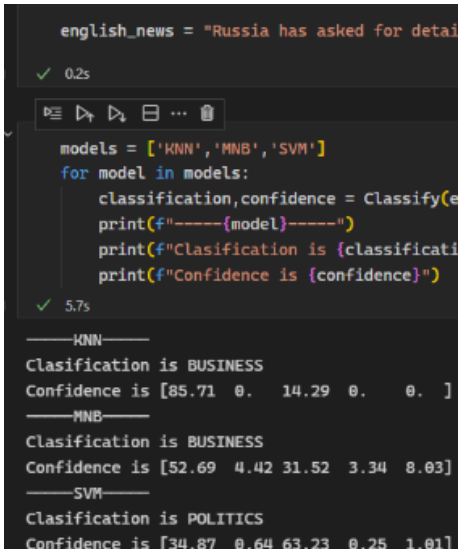


Figure 5.15: Performance Test for English News

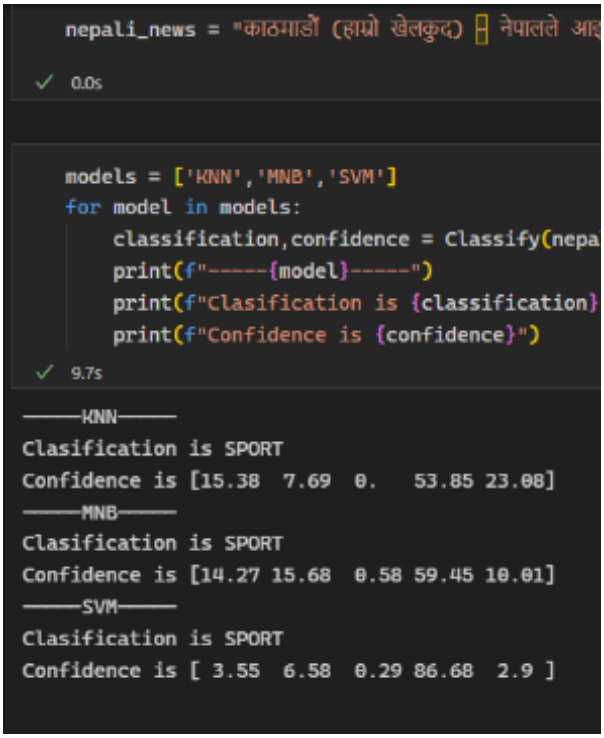


Figure 5.16: Performance Test for Nepali News

The Figures 5.15 and 5.16 above shows that the models classified the English News within 5.7s and Nepali News withing 9.7s. Hence passing the performance test.

5.2.2 System Testing

System testing is a process of verifying that a software system meets the specified requirements and works as intended. It evaluates the system as a whole and ensures its correct functioning.

Table 5.4: Test Case for Classification and Summarization

S. N	Test Cases	Test Description	Input Test Data	Expected Results	Actual Result	Remarks
1.	TC-IVS-1	Handle different types of inputs, such as text data of varying length and format.	English News	Classified English News	Classified English News	Pass

2.	TC-IVS-2	Handle different types of inputs, such as text data of varying length and format.	Nepali News	Classified Nepali News	Classified Nepali News	Pass
3.	TC-PS-1	System performs efficiently under normal and peak loads.	English News Article of varying lengths.	Classifies and summarizes the news under minimal time.	Classified the news under 30 secs.	Pass
4.	TC-PS-2	System performs efficiently under normal and peak loads.	Nepali News Article of varying lengths.	Classifies and summarizes the news under minimal time.	Classified the news under 30 secs.	Pass

The Table 5.4 above shows the test cases for whole system done at once. All possible tests were performed and every test case as written in Table 5.4 produced satisfactory results.

Table 5.5: Test Cases for Scraping

S. N	Test Cases	Test Description	Input Test Data	Expected Results	Actual Result	Remarks
1.	TC-SS-1	Extract the relevant information from the HTML content	News Portal Links	Scrapped News, Title, Link from the site	Scrapped News, Title, Link	Pass

2.	TC-SS-2	Display Summarized News.	-	Display Summarized news, title, date and source link	Summarized news, title, date and source link displayed	Pass

The above Table 5.5 are the test cases for scraping module. For scraping, Ekantipur, Setopati for Nepali news and BBC, The Kathmandu Post for English news were the portals. The systems successfully scraped the news articles along with the title and date and was presented to the user with the summary.

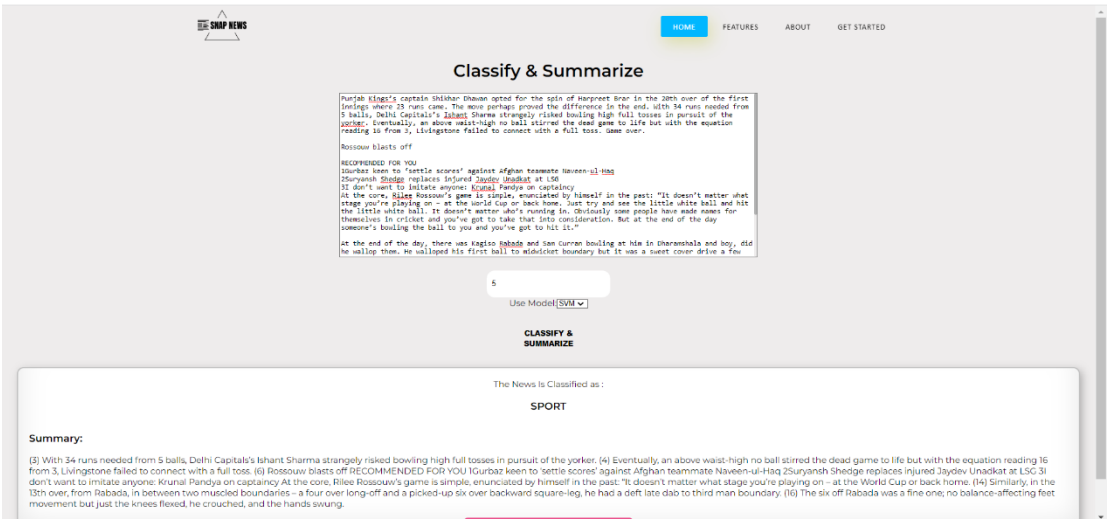


Figure 5.17: System Testing for English News

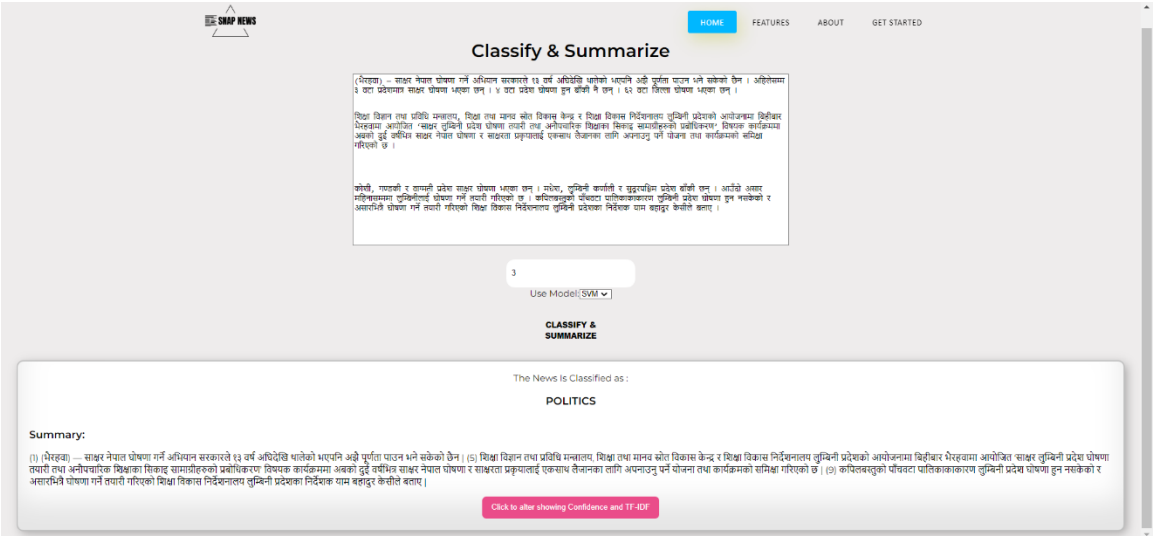


Figure 5.18: System Testing for Nepali News

5.3 Result Analysis

The system was tested through unit testing and proved to be effective in executing its intended functions. The results showed that the project was able to meet its goals, but there is still room for improvement in terms of expanding the system's capabilities and increasing community involvement.

5.3.1 Evaluating Accuracy

In machine learning, accuracy is a common metric used to evaluate the performance of a classifier model. Accuracy measures the proportion of correctly classified instances among all instances in the dataset. To calculate accuracy, the first step is to divide the dataset into two parts: a training set and a test set. The training set is used to train the model, while the test set is used to evaluate the model's performance.

In classifier model the most common measure to evaluate accuracy are:

- Precision: Precision is the fraction of true positives among all the positive predictions made by the model. It measures how accurate the model is when predicting positive instances. The formula for precision is:
  Precision = True Positives / (True Positives + False Positives)
- Recall: Recall is the fraction of true positives among all the actual positive instances in the dataset. It measures how well the model is able to identify positive instances. The formula for recall is:
  Recall = True Positives / (True Positives + False Negatives)
- F1 score: The F1 score is the harmonic mean of precision and recall. It provides a single score that balances the tradeoff between precision and recall. The F1 score ranges from 0 to 1, where a score of 1 represents perfect precision and recall, and 0 represents the worst performance. The formula for F1 score is:
  F1 score = 2 * (Precision * Recall) / (Precision + Recall)

The most common scoring metrics for summarization are:

- ROUGE-1: ROUGE-1 calculates the unigram overlap between the generated summary and the reference summary. The recall is calculated by dividing the number of overlapping unigrams in the generated summary by the total number of unigrams in the reference summary. Precision is calculated by dividing the number

of overlapping unigrams in the generated summary by the total number of unigrams in the generated summary. The harmonic mean of recall and precision is then used to calculate the F1-score.

- ROUGE-2: ROUGE-1 calculates the unigram overlap between the generated summary and the reference summary. The recall is calculated by dividing the number of overlapping bigrams in the generated summary by the total number of bigrams in the reference summary. Precision is calculated by dividing the number of overlapping bigrams in the generated summary by the total number of bigrams in the generated summary. The harmonic mean of recall and precision is then used to calculate the F1-score.
- ROUGE-L: ROUGE-L measures the longest common subsequence (LCS) between the generated summary and the reference summary,

5.3.1.1 Evaluating Accuracy of KNN

For English Dataset:

```
Train Accuracy Score : 97
Test Accuracy Score  : 97

               precision    recall  f1-score   support

     business       0.97      0.97      0.97        64
entertainment       0.92      1.00      0.96        54
     politics       0.97      0.92      0.94        64
        sport       1.00      1.00      1.00        61
         tech       0.98      0.95      0.96        55

     accuracy                           0.97       298
    macro avg       0.97      0.97      0.97       298
 weighted avg       0.97      0.97      0.97       298
```

Figure 5.19: Classification Report of KNN for English Dataset

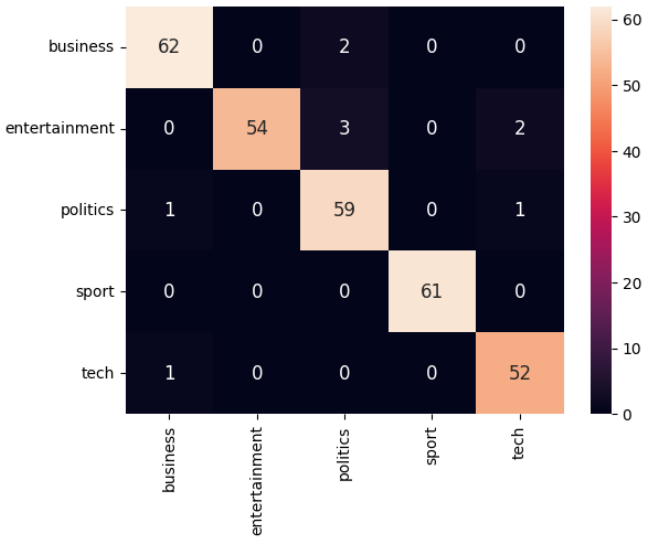


Figure 5.20: Confusion Matrix of KNN for English Dataset

For English Dataset the best hyperparameter was found to be n_neighbors = 7 and metric = cosine. The accuracy was 97% for both training and test dataset. Sport news were perfectly classified. From confusion matrix it is seen that politics was sometimes misclassified as entertainment and business. The Falsest Positive was for class Entertainment and most False Negative for Tech. The model performs well with 97% macro-average and weighted-average f1-score.

For Nepali Dataset:

```
Train Accuracy Score : 92
Test Accuracy Score  : 90

              precision    recall  f1-score   support

     business       0.92      0.85      0.88       218
entertainment       0.83      0.94      0.88       192
     politics       0.91      0.89      0.90       102
        sport       0.96      0.92      0.94       191
         tech       0.91      0.91      0.91       205

     accuracy                           0.90       908
    macro avg       0.90      0.90      0.90       908
 weighted avg       0.90      0.90      0.90       908
```

Figure 5.21: Classification Report of KNN for Nepali Dataset

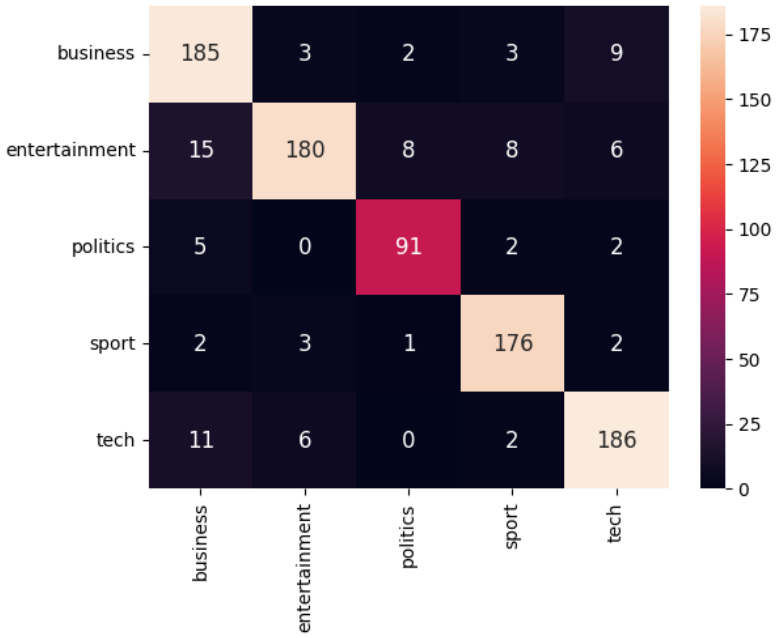


Figure 5.22: Confusion Matrix of KNN for Nepali Dataset

For Nepali dataset the best hyperparameter for KNN was n_neighbors = 13 and metric = cosine. The Training accuracy was 92% and test accuracy was 90%. The Falsest Positive classification was for Entertainment. Business class had the Falsest Negatives. There were more False Positives and False Positive compared to English Dataset. The model is performing quite decent with 90% macro-average and weighted average f1-score.

5.3.1.2 Evaluating Accuracy of MNB

For English Dataset

```
Train Accuracy Score : 99
Test Accuracy Score  : 96

               precision    recall  f1-score   support

     business       1.00      0.90      0.95        71
entertainment       0.90      1.00      0.95        53
     politics       0.90      0.96      0.93        57
        sport       1.00      1.00      1.00        61
         tech       0.98      0.93      0.95        56

     accuracy                           0.96       298
    macro avg       0.96      0.96      0.96       298
 weighted avg       0.96      0.96      0.96       298
```

Figure 5.23: Classification Report of MNB for English Dataset

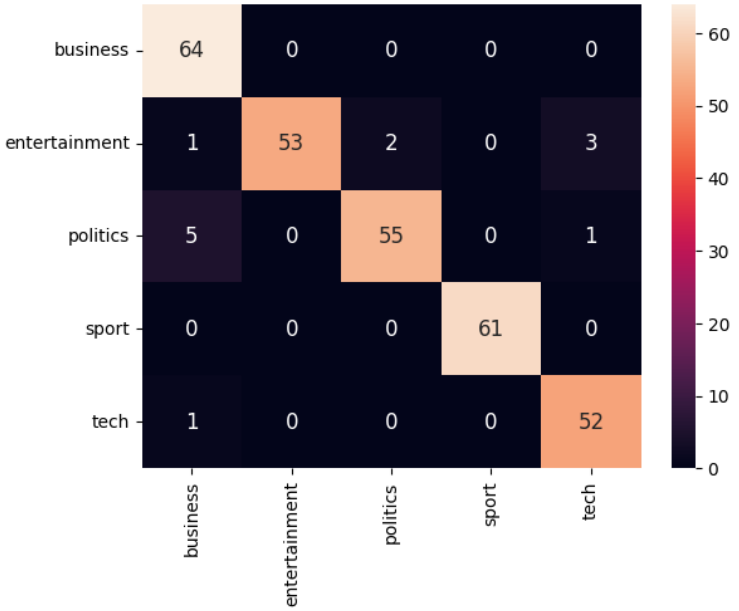


Figure 5.24: Confusion Matrix of MNB for English Dataset

For English Dataset the best hyperparameter for MNB was fit_prior = True with alpha = 1. The training accuracy was found to be 99% and test accuracy was 96%. This model has

most False Positives for Entertainment class. Business class had the Falsest Negatives. However, it performed little worse than KNN with 96% accuracy on macro-average and weighted-average scores f1-score. KNN had performed well with 97% accuracy.

For Nepali Dataset

```
Train Accuracy Score : 93
Test Accuracy Score  : 91

               precision    recall  f1-score   support

     business       0.96      0.81      0.88       239
entertainment       0.98      0.87      0.92       244
     politics       0.63      1.00      0.77        63
        sport       0.95      0.98      0.96       177
         tech       0.87      0.97      0.92       185

     accuracy                           0.91       908
    macro avg       0.88      0.93      0.89       908
 weighted avg       0.92      0.91      0.91       908
```

Figure 5.25: Classification Report of MNB for Nepali Dataset

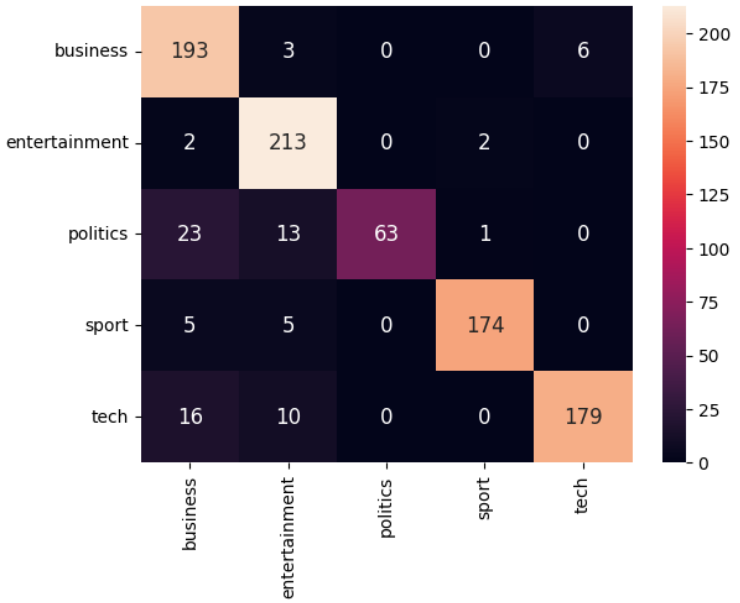


Figure 5.26: Confusion Matrix of MNB for Nepali Dataset

For Nepali Dataset the best hyperparameter for MNB was also fit_prior = True with alpha = 1. The training accuracy was found to be 93% and test accuracy was 91%. This model

has most False Positives for Politics class. Business class had the Falsest Negatives. However, it performed little better than KNN with 91% accuracy weighted-average score f1-score. KNN had performed little off with 90% accuracy. However, it scores 89% on macro-average score f1-score compared to KNN that had 90%.

5.3.1.3 Evaluating Accuracy of SVM

For English Dataset

```
Train Accuracy Score : 100
Test Accuracy Score  : 98

               precision    recall  f1-score   support

     business       1.00      0.98      0.99        65
entertainment       0.97      0.98      0.97        58
     politics       0.95      0.98      0.97        59
        sport       1.00      1.00      1.00        61
         tech       1.00      0.96      0.98        55

     accuracy                           0.98       298
    macro avg       0.98      0.98      0.98       298
 weighted avg       0.98      0.98      0.98       298
```

Figure 5.27: Classification Report of SVM for English Dataset

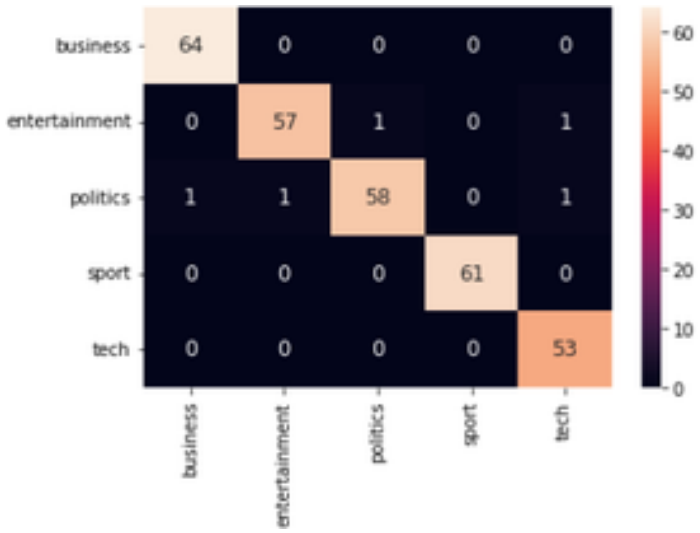


Figure 5.28: Confusion Matrix of SVM for English Dataset

For English Dataset best SVM hyperparameter was found to be C =1, gamma = 1 and kernel = linear. The train accuracy was 100% and test accuracy was found to be 98%. Tech had

the Falsest Negatives. Politics was falsely positive the most. The SVM model performed best among KNN and MNB. The macro-average and weighted-average f1-score were 98%.

For Nepali Dataset

```
Train Accuracy Score : 100
Test Accuracy Score  : 95

              precision    recall  f1-score   support

     business       0.92      0.92      0.92       202
entertainment       0.97      0.94      0.95       224
     politics       0.95      0.98      0.96        97
        sport       0.96      0.99      0.98       179
         tech       0.94      0.93      0.93       206

     accuracy                           0.95       908
    macro avg       0.95      0.95      0.95       908
 weighted avg       0.95      0.95      0.95       908
```

Figure 5.29: Classification Report of SVM for Nepali Dataset

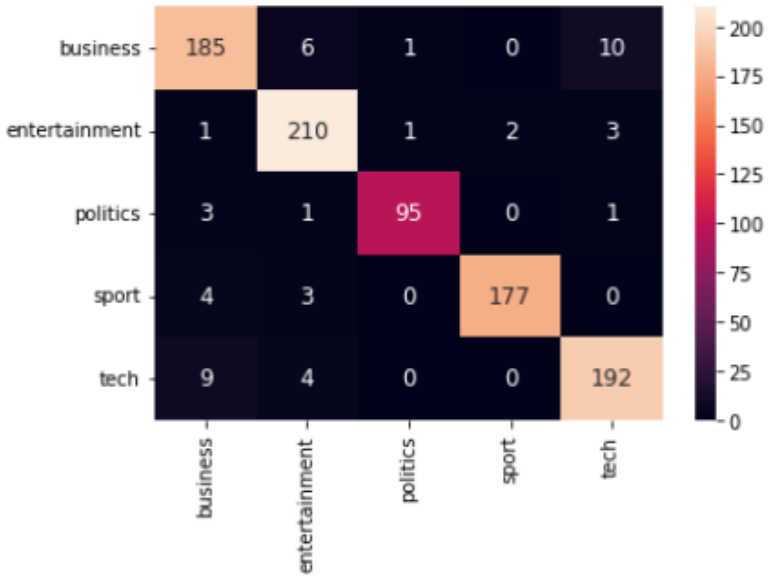


Figure 5.30: Confusion Matrix of SVM for Nepali Dataset

For Nepali Dataset the best hyperparameter was C = 10, gamma = 0.1 and kernel = rbf, the training accuracy was 100% and test accuracy was 95%. Business was falsely positive and

falsely negative the most. The SVM model performed best with weighted-average and macro average f1-score of 95%.

Thus, SVM performed best for both English and Nepali dataset. So SVM model was saved and deployed for web application.

5.3.1.4 Evaluating ROUGE score for summarization

For English Dataset

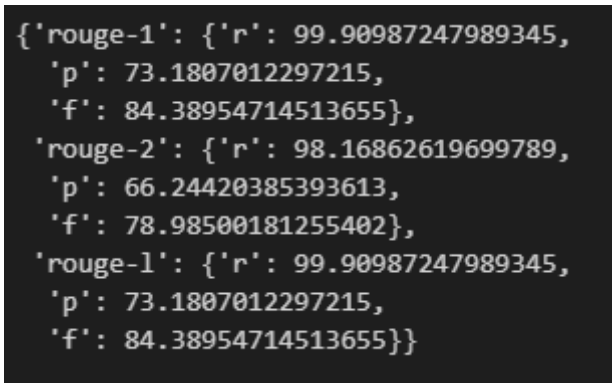


Figure 5.31: ROUGE score with respect to actual news

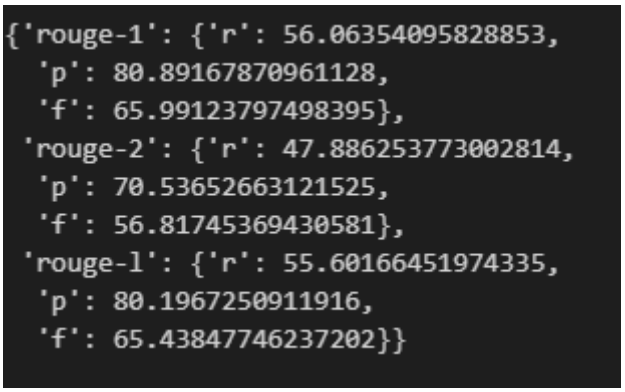


Figure 5.32: ROUGE score with respect to actual summary

For English Dataset the actual summary was on average 45% sentences of the news. So, summaries were generated to 45% sentences of the news.

ROUGE score of generated extractive summary often performs well with respect to whole text (here news). The generated summary all lies in the main texts. So, recall is 99% for all ROUGE-1, ROUGE-2 and ROUGE-l when compared to news. While precision is a little low. Nevertheless, when compared with news f1-scores are quite decent as it should be for

an extractive summary. The f1-scores are 84%, 78% and 84% respectively. This shows unigrams were preserved well in generated summary when compared to bigrams. While Longest Common Sequence was decent to.

However, score of generated summaries with actual or reference summary was a bit less as expected. Recall score is little low as all words in actual summary may not be in generated summary. But precision score was decent with 80%,70% and 80% of ROUGE-1, ROUGE-2 and ROUGE-l respectively. Thus, f1-score of 65%,56% and 65% was observed respectively.

For Nepali Dataset

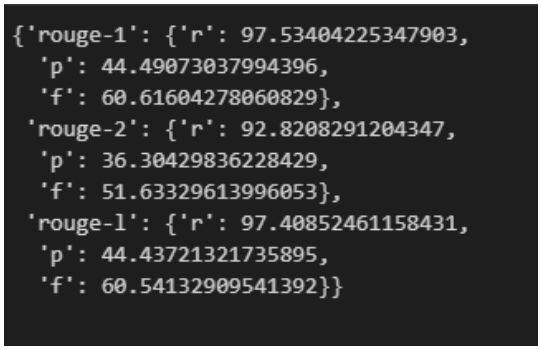


Figure 5.33: ROUGE score with respect to actual news

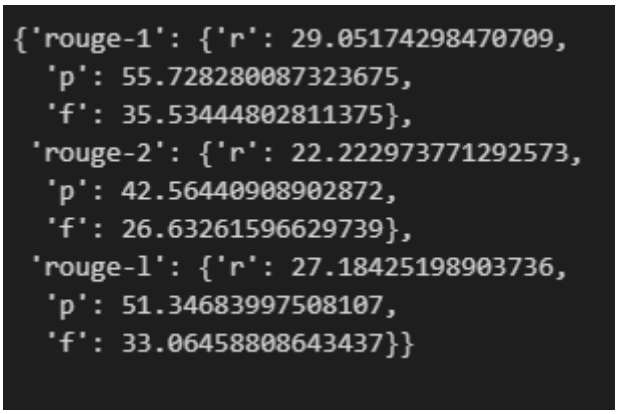


Figure 5.34: ROUGE score with respect to actual summary

For Nepali Dataset the actual summary was on average 17% sentences of the news. So, summaries were generated to 17% sentences of the news.

ROUGE score of generated extractive summary often performs well with respect to whole text (here news). The generated summary all lies in the main texts. So, recall is 97%,92% and 97% for ROUGE-1, ROUGE-2 and ROUGE-l respectively when compared to news.

But precision is very low. This is also because the 17% summarization of whole text is very low. Due to low precision f1-score is also quite low. The f1-scores are thus 60%,51% and 60% respectively.

The ROUGE scores when compared to actual summary shows poor performance for Nepali Dataset. Part of it is because the summarized contents are only 17% of actual news. The f1-scores are 35%, 26% and 33%. This can be considered as relatively low performance.

CHAPTER 6
CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 Conclusion

In conclusion, The SnapNews is able to classify English and Nepali news into five categories business, entertainment, politics, sports and tech. Since SVM produced the highest accuracy amongst the models trained. SVM was used as the model for the system. The model trained is able to categorize the news accurately based on their content which can also help users to quickly find news that is relevant to their interests.

Furthermore, the summarizer uses an extractive summarize algorithm that identifies the most important sentences in both English and Nepali news articles using the TF-IDF values. This can help users in understanding the news in a short amount of time without spending most of their time reading the whole news article.

Finally, SnapNews uses BeautifulSoup that can collect news articles from the multiple sources, classify them and then summarize the content which helps the users to stay up-to-date with the latest news without having to manually search for articles.

Overall, the project can classify, summarize and scrape the news articles in English and Nepali which can save time and effort for users, and can help them to stay up-to-date with the latest news in their field of interest.

6.2 Future Recommendations

There are several potential future recommendations that could be implemented in the Snap News project:

- **Multilingual support:** Currently, the system only supports news articles in English and Nepali. Adding support for multiple languages could make the system more accessible to a wider range of users.
- **Personalization:** Adding personalized news recommendations based on user preferences and reading history could improve the user experience and increase engagement with the system.

- **Sentiment analysis:** Adding sentiment analysis to the news classification and summarization tasks could provide users with a better understanding of the tone and emotion of the article's content.
- **Real-time updates:** Implementing real-time updates could enable users to receive breaking news alerts and stay up-to-date on current events as they happen.
- **Mobile app:** Developing a mobile app could make the system more accessible and convenient for users who prefer to consume news on their mobile devices.
- **Social media integration:** Integrating with social media platforms could allow users to share news articles and engage with other users who are interested in similar topics.

By implementing these future recommendations, the Snap News project could continue to evolve and improve, providing a more valuable and engaging service to its users.

REFERENCES

[1] I. Sommerville, "Incremental Delivery," in *Software Engineering*, Harlow, Pearson Education Limited , 2016, p. 64.

[2] T. Joachims, " Text categorization with support vector machines: Learning with many relevant features," *European conference on machine learning,* vol. 1398, pp. 137-142, 2005.

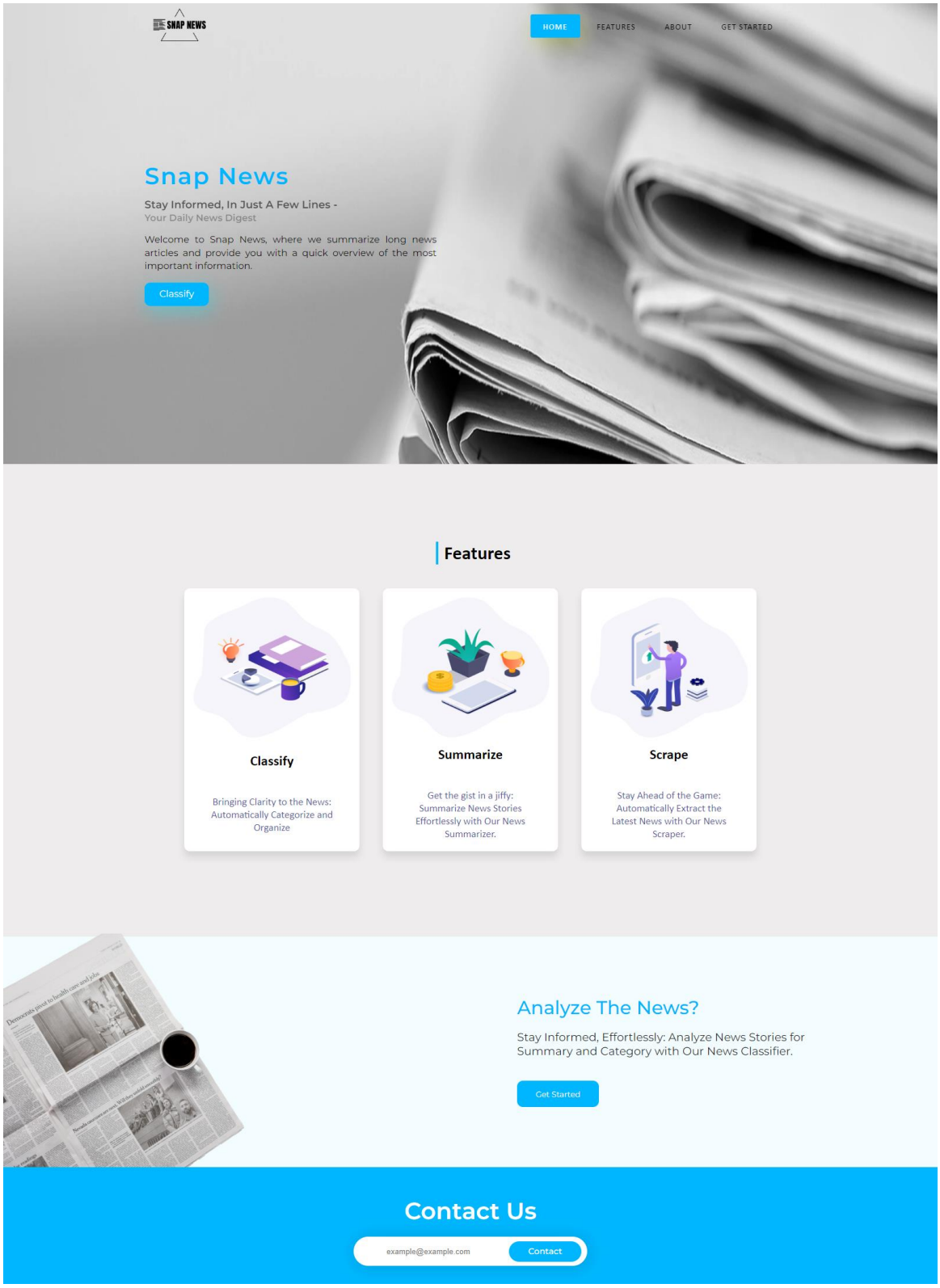
[3] G. a. R. S. B. a. S. K. Krishnalal, "A new text mining approach based on HMM-SVM for web news classification," *International Journal of Computer Applications,* vol. 1, pp. 98-104, 2010.

[4] T. F. a. H. M. B. a. F. Z. T. Gharib, "Arabic Text Classification Using Support Vector Machines.," *Int. J. Comput. Their Appl.,* vol. 16, no. 4, pp. 192-199, 2009.

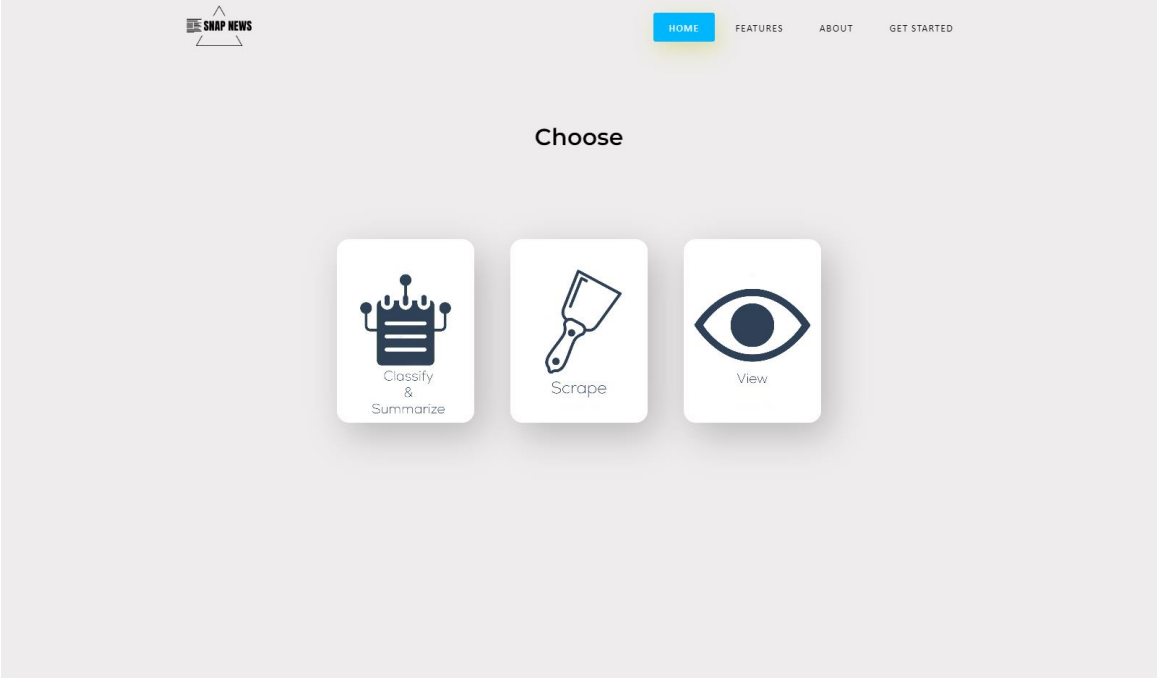
[5] K. a. M. K. N. Raghuveer, "Text Categorization in Indian Languages using Machine Learning Approaches.," in *IICAI*, 2007, pp. 1864-1883.

APPENDICES

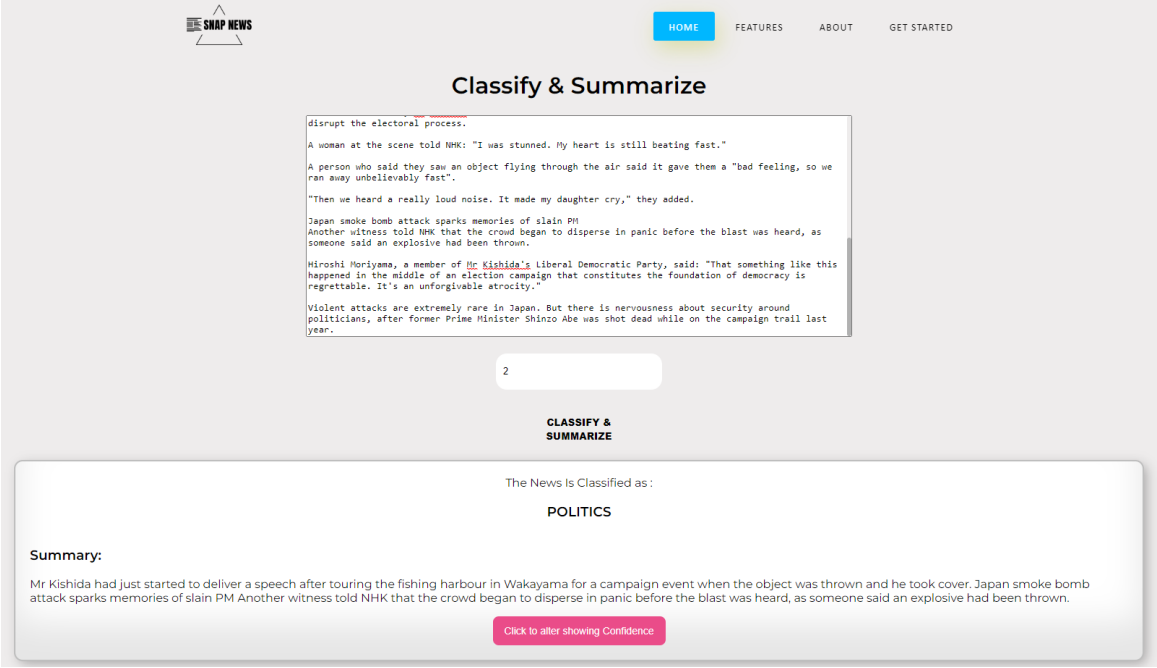
(Screenshots)



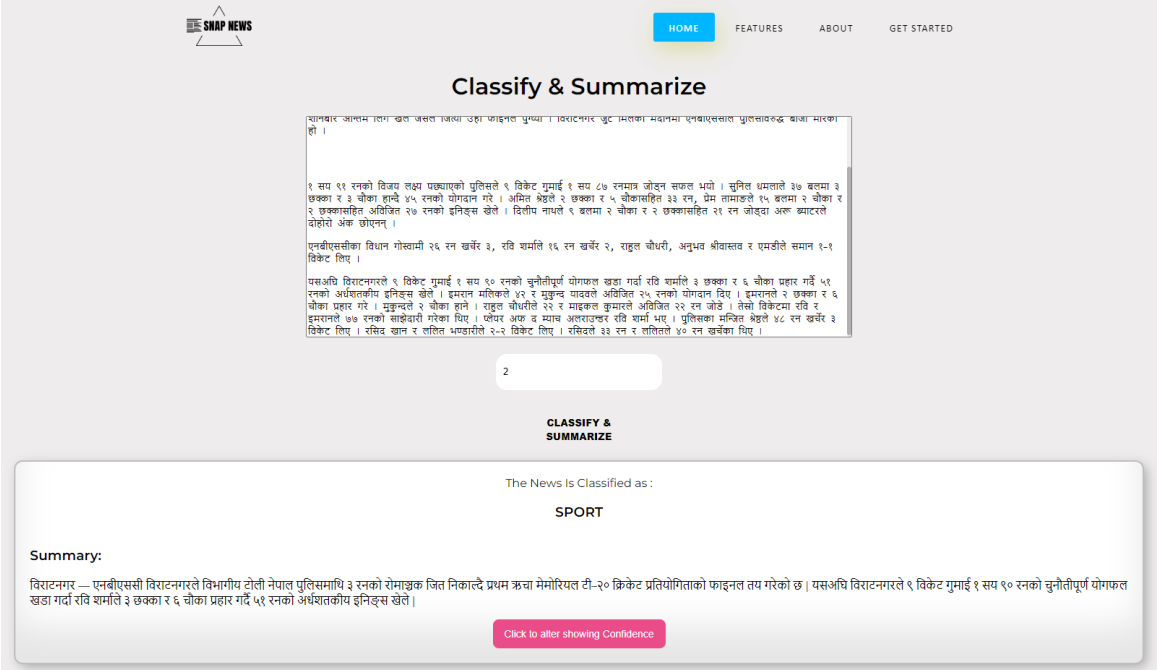
Home Page



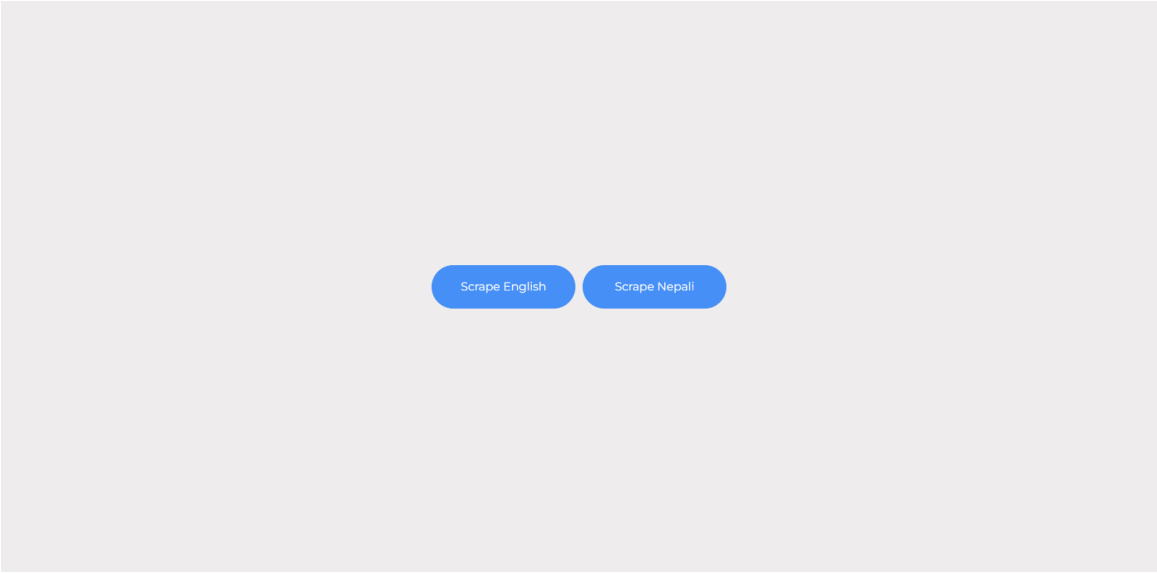
Get Started Page



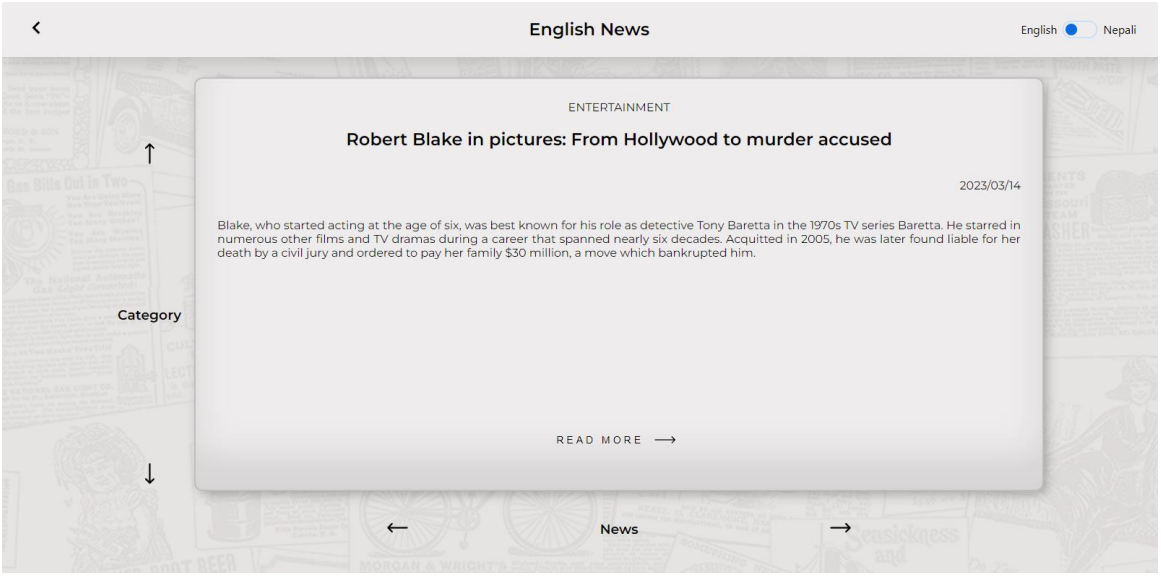
English News Classify and Summarize



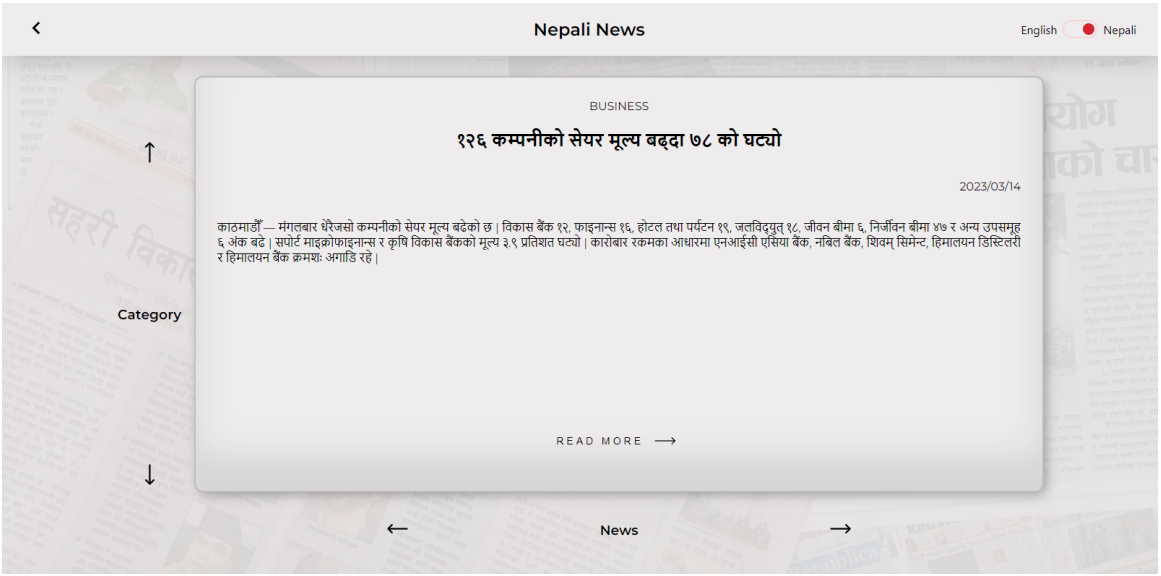
Nepali News Classify and Summarize



Scrape Screen



English News Scraped



Nepali News Scraped