National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Rutuja Anil Pande
Student ID: x21239444

School of Computing
National College of Ireland

Supervisor: Dr.Ahmed Makki

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Rutuja Anil Pande
Student ID:	x21239444
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr.Ahmed Makki
Submission Due Date:	14/08/2023
Project Title:	Configuration Manual
Word Count:	1042
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	13th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rutuja Anil Pande
x21239444

1 Introduction

The proposed model aims to generate short abstractive text summaries from long documents and perform sentimental analysis on that generated summaries. Additionally, the English summaries which are generated are then translated in Hindi language. The purpose of configuration manual is to outline the software, hardware requirements of this project and guide for any implementation if needed. All the stages of implementation of the research is discussed here in configuration manual starting from data transformation, model building, and testing and validation of the models.

2 System Configuration

The need of Hardware and Software for this research implemented are discussed in detail in this section.

2.1 Hardware Requirements

The hardware specifications used in this research is the machine in which the the project is implemented in. The computer is Mac book Air M2 Chip with 8-core CPU, 10-core GPU and 8GB RAM and 512 GB storage. The below figure shows the information about the system used.

Figure 1: Hardware Specifications

2.2 Software Requirements

The implementation of the research is done on Google Colab platform, as it gives a decent amount of memory and GPU for the execution of the code. Both the datasets have a large number of records as corpus and requires a powerful resources for efficient computation. With Google Colab, this study was able to perform the code with the capacity of the resources provided. The CSV file dataset of Indian News Summary is uploaded on the Google Colab IDE and the other dataset BBC News Summary which contains directories for news articles and summaries are uploaded directly on google drive and the they are mounted on google colab. Python language is used for this research. Fig 2. shows the mounting of drive to google colab.

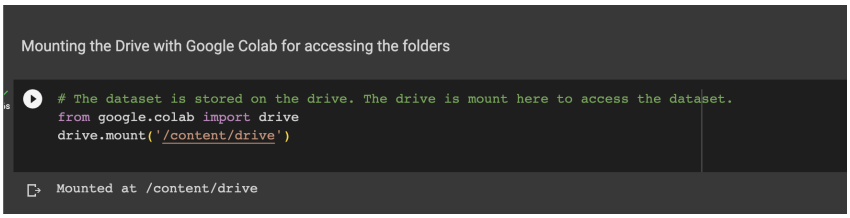


Figure 2: Google Drive mounted on Google Colab

2.3 Code Execution

The code can be run on jupyter notebook or google colab. For this study, google colab is used for running the code. Google Colab Pro was subscribed for the implementation of the project. The dataset can be directly upload on the colab platform or can be uploaded on the drive and then mounted on colab.

3 Importing required Python Packages

There are different packages and libraries in python which needs to be defined or imported before using this libraries like re, nlkt, pandas, numpy, matplotlib and so on. These are downloaded for every step of code implementation. The Fig. 3 shows all the libraries and packages imported.

4 Exploratory Data Analysis

EDA steps are carried out to observe the patterns or any imbalance in the datasets. The visualization of bar plots shows these patterns such as in the case of this study, The visualizations are done to see the varying lengths of the texts in the dataset. Outliers are identified and then removed if necessary from the dataset. Below Figures shows some of the visual representation of the data and the patterns in it.

5 Data Collection & Data Transformation

The Data collection pre-processing steps and transformation is discussed in this section.

```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
import nltk.corpus
nltk.download('stopwords')
from nltk.corpus import stopwords
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
nltk.download('omw-1.4')
!pip install TextBlob
from textblob import TextBlob
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from collections import Counter
!pip install spacy
import spacy
! pip install nltk
! python -m spacy download en_core_web_sm
! pip install transformers
import tensorflow as tf
from transformers import pipeline
from transformers import BertTokenizer, TFBertModel
```

Figure 3: Libraries Imported

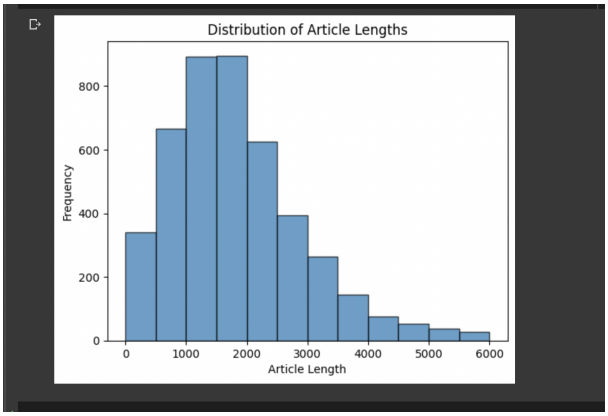


Figure 4: Distribution of Article Lengths for Indian Dataset

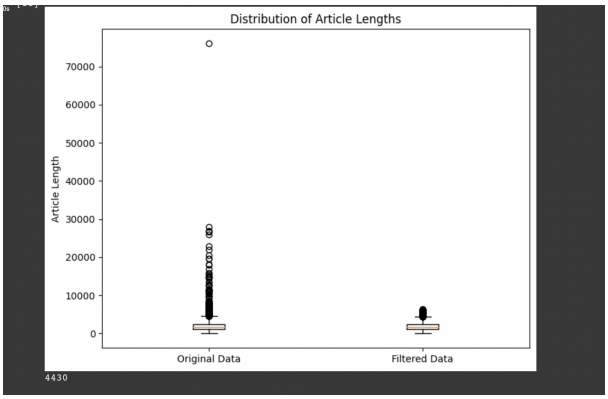


Figure 5: Outliers in Indian News Dataset

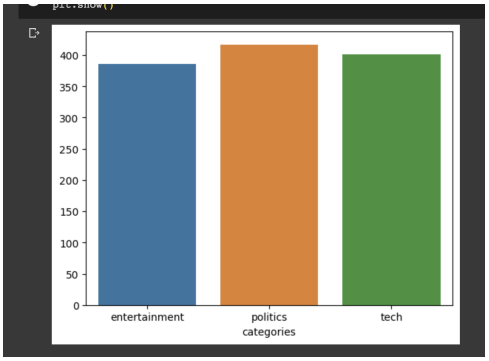


Figure 6: News Articles count in Categories

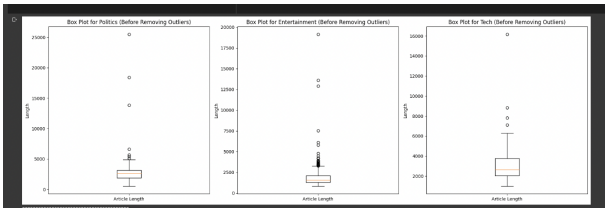


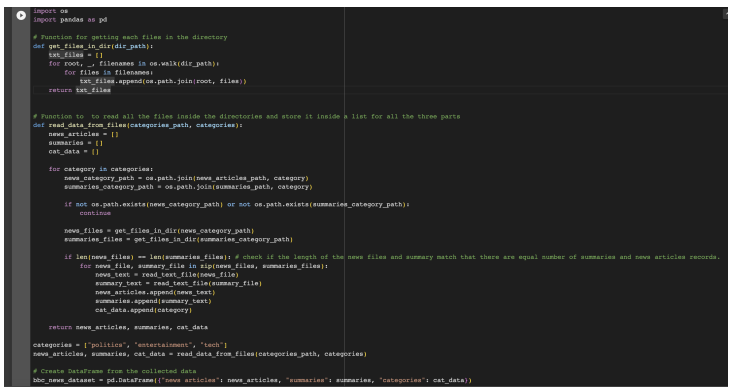
Figure 7: Outliers in BBC News Dataset

5.1 Data Collection

The data used in this study is downloaded from the kaggle website and the link is given below. One dataset is CSV file with 4433 rows and 6 columns and the other dataset is a text files contained inside the folders.

- Dataset 1 : https://www.kaggle.com/datasets/sunnysai12345/news-summary
- Dataset 2 : https://www.kaggle.com/datasets/pariza/bbc-news-summary

The Fig 4 . shows the collection of data for the second dataset - BBC News Summary.



```
import os
import pandas as pd

# Function for getting each files in the directory
def get_files_in_dir(dir_path):
    txt_files = []
    for root, _, filenames in os.walk(dir_path):
        for files in filenames:
            txt_files.append(os.path.join(root, files))
    return txt_files


# Function to  to read all the files inside the directories and store it inside a list for all the three parts
def read_data_from_files(categories_path, categories):
    news_articles = []
    summaries = []
    cat_data = []

    for category in categories:
        news_category_path = os.path.join(news_articles_path, category)
        summaries_category_path = os.path.join(summaries_path, category)

        if not os.path.exists(news_category_path) or not os.path.exists(summaries_category_path):
            continue

        news_files = get_files_in_dir(news_category_path)
        summaries_files = get_files_in_dir(summaries_category_path)

        if len(news_files) == len(summaries_files): # check if the length of the news files and summary match that there are equal number of summaries and news articles records.
            for news_file, summary_file in zip(news_files, summaries_files):
                news_text = read_text_file(news_file)
                summary_text = read_text_file(summary_file)
                news_articles.append(news_text)
                summaries.append(summary_text)
                cat_data.append(category)

    return news_articles, summaries, cat_data

categories = ["politics", "entertainment", "tech"]
news_articles, summaries, cat_data = read_data_from_files(categories_path, categories)

# Create DataFrame from the collected data
bbc_news_dataset = pd.DataFrame({"news articles": news_articles, "summaries": summaries, "categories": cat_data})
```

Figure 8: Data Collection for BBC News Summary

5.2 Data Pre-Processing

The pre-processing steps included in the study is to remove some special characters, punctuation, spaces or any html tags from the text. For better readability the text is also converted to lower case. Fig 5,6. shows the pre-processing steps carried out for the data.

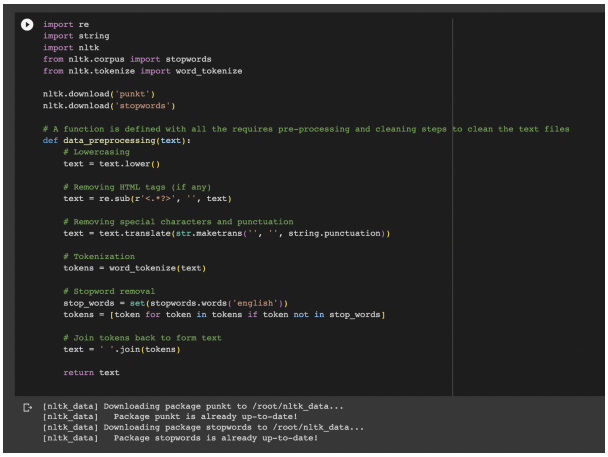


Figure 9: Preprocessing for Dataset 1

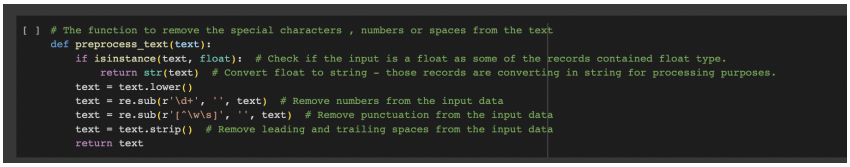


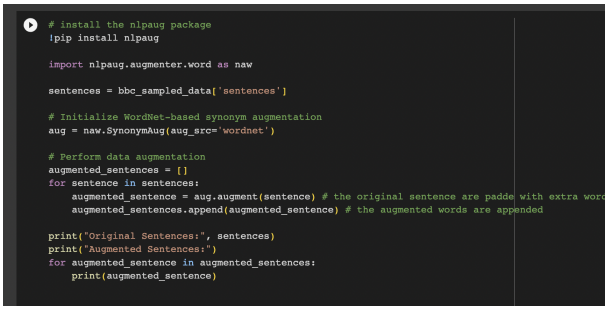
Figure 10: Preprocessing for Dataset 2

5.3 Data Transformation

Data transformation is done on the basis of the length of text. If the data was imbalance then data augmentation is performed to increase the size of text and to improve the quality of summaries. The data also had some outliers. Therefore, data was filtered by removing those outliers. These data transformation techniques are used in the study to improve the quality of the performance of models. Fig 7,8. shows the data transformation done for smooth execution on datasets.

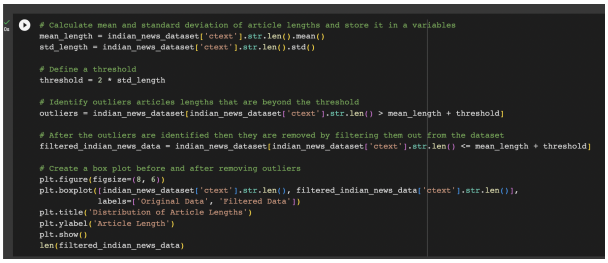
6 Model Building

For implementing the objective of the project, models are build and executed on datasets to generate abstractive text summaries, sentimental analysis, and translation from English to Hindi language. Different models are used for satisfying the objectives like BART, BERT and Google Translator.



```
# install the nlpaug package
!pip install nlpaug

import nlpaug.augmenter.word as naw

sentences = bbc_sampled_data['sentences']

# Initialize WordNet-based synonym augmentation
aug = naw.SynonymAug(aug_src='wordnet')

# Perform data augmentation
augmented_sentences = []
for sentence in sentences:
    augmented_sentence = aug.augment(sentence) # the original sentence are padde with extra word
    augmented_sentences.append(augmented_sentence) # the augmented words are appended

print("Original Sentences:", sentences)
print("Augmented Sentences:")
for augmented_sentence in augmented_sentences:
    print(augmented_sentence)
```

Figure 11: Data augmentation

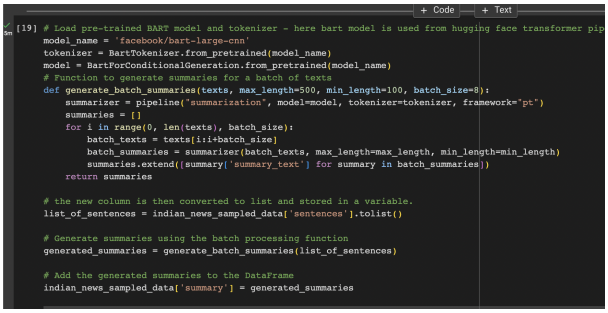


```
# Calculate mean and standard deviation of article lengths and store it in a variables
mean_length = indian_news_dataset['ctext'].str.len().mean()
std_length = indian_news_dataset['ctext'].str.len().std()

# Define a threshold
threshold = 2 * std_length

# Identify outliers articles lengths that are beyond the threshold
outliers = indian_news_dataset[indian_news_dataset['ctext'].str.len() > mean_length + threshold]

# After the outliers are identified then they are removed by filtering them out from the dataset
filtered_indian_news_data = indian_news_dataset[indian_news_dataset['ctext'].str.len() <= mean_length + threshold]

# Create a box plot before and after removing outliers
plt.figure(figsize=(8, 6))
plt.boxplot([indian_news_dataset['ctext'].str.len(), filtered_indian_news_data['ctext'].str.len()],
            labels=['Original Data', 'Filtered Data'])
plt.title('Distribution of Article Lengths')
plt.ylabel('Article Length')
plt.show()
len(filtered_indian_news_data)
```

Figure 12: Data Filteration

6.1 BART Model

BART is a pre-trained model used for generating abstractive text summaries from the datasets. The parameters of min, max length and batch size is passed for processing the execution. Fig 9. shows the implementation of BART model.



```
+ Code  + Text
[19] # Load pre-trained BART model and tokenizer - here bart model is used from hugging face transformer pip
model_name = 'facebook/bart-large-cnn'
tokenizer = BartTokenizer.from_pretrained(model_name)
model = BartForConditionalGeneration.from_pretrained(model_name)
# Function to generate summaries for a batch of texts
def generate_batch_summaries(texts, max_length=500, min_length=100, batch_size=8):
    summarizer = pipeline("summarization", model=model, tokenizer=tokenizer, framework="pt")
    summaries = []
    for i in range(0, len(texts), batch_size):
        batch_texts = texts[i:i+batch_size]
        batch_summaries = summarizer(batch_texts, max_length=max_length, min_length=min_length)
        summaries.extend([summary['summary_text'] for summary in batch_summaries])
    return summaries

# the new column is then converted to list and stored in a variable.
list_of_sentences = indian_news_sampled_data['sentences'].tolist()

# Generate summaries using the batch processing function
generated_summaries = generate_batch_summaries(list_of_sentences)

# Add the generated summaries to the DataFrame
indian_news_sampled_data['summary'] = generated_summaries
```

Figure 13: BART Model

6.2 BERT Model

BERT model is also a pre-trained model which has various variants of it. In this study, distilbert as a variant is implemented for sentimental analysis task as bert is pre-trained on contextual data and can represent data contextual format with associated semantics by defineing the resltionship in the sentences. Fig 10. shows the implementation of BERT model.

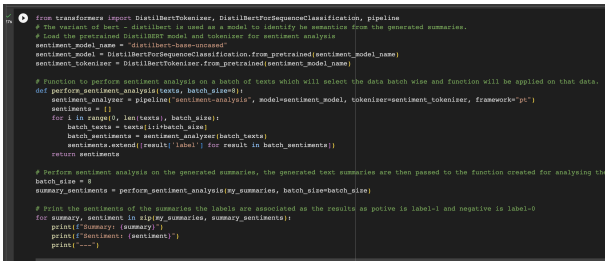


Figure 14: BERT Model

6.3 Google Translator

The google translator is a tried and tested model for translation purpose from one language to another. The google translator can be used for multilingual translation, speech-to-text translation and so on. Fig 11. shows the translation of English text to Hindi Text.

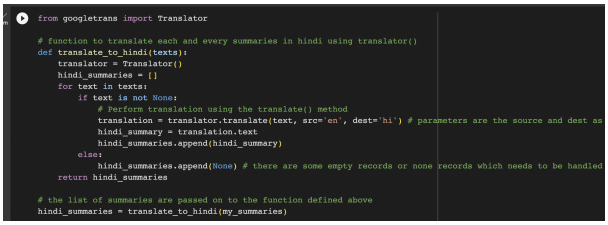


Figure 15: Google Translator

6.4 Evaluation Metrics

The model applied on the dataset and the output generated are measured using evaluation metric models like ROUGE score amd BLEU score. ROUGE score is calculated for the summaries to check the quality and the BLEU score is used for measuring the translation. Fig 12,13. shows the implementation of evaluation metric models.

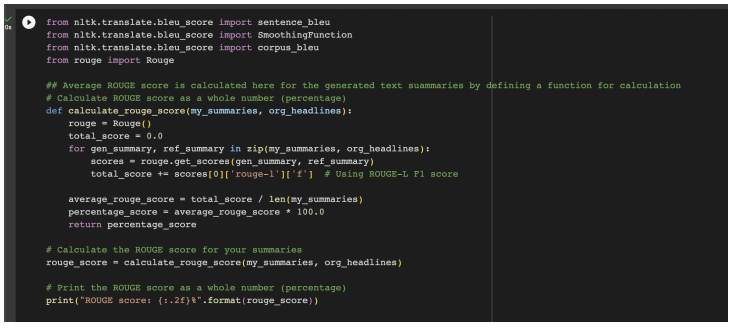
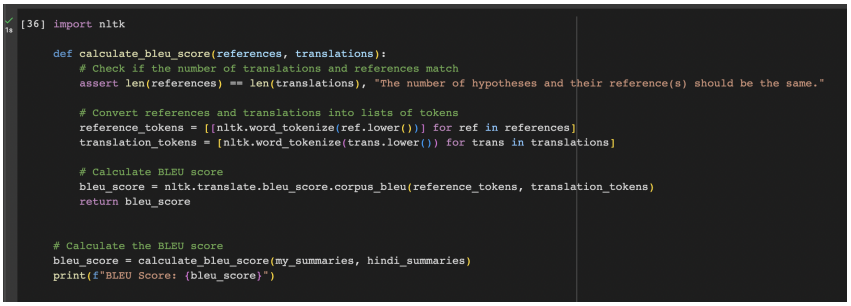


Figure 16: ROUGE score for Text Summaries

https://huggingface.co/docs/transformers/model$_doc$/*bart*
https://huggingface.co/docs/transformers/model$_doc$/*distilbert*
https://stackabuse.com/text-translation-with-google-translate-api-in-python/: :text=You%20can%2



```
[36] import nltk

    def calculate_bleu_score(references, translations):
        # Check if the number of translations and references match
        assert len(references) == len(translations), "The number of hypotheses and their reference(s) should be the same."

        # Convert references and translations into lists of tokens
        reference_tokens = [[nltk.word_tokenize(ref.lower())] for ref in references]
        translation_tokens = [nltk.word_tokenize(trans.lower()) for trans in translations]

        # Calculate BLEU score
        bleu_score = nltk.translate.bleu_score.corpus_bleu(reference_tokens, translation_tokens)
        return bleu_score


    # Calculate the BLEU score
    bleu_score = calculate_bleu_score(my_summaries, hindi_summaries)
    print(f"BLEU Score: {bleu_score}")
```

Figure 17: BLEU score for translation

References