**BATCH:** A

**DIV:** D17B

**SUBJECT:** NLP

**TOPIC:** Summarizing News Article

**GROUP MEMBERS:**

Sanjana Asrani 01

Abhishek Chhabria 15

Jayesh Dayalani 20

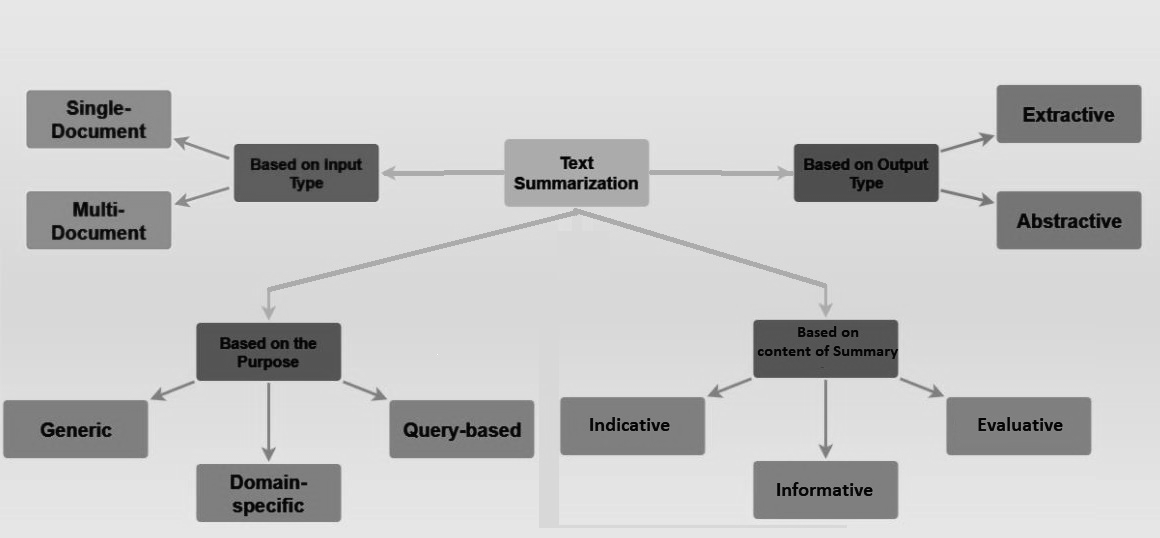
**Aim :-**

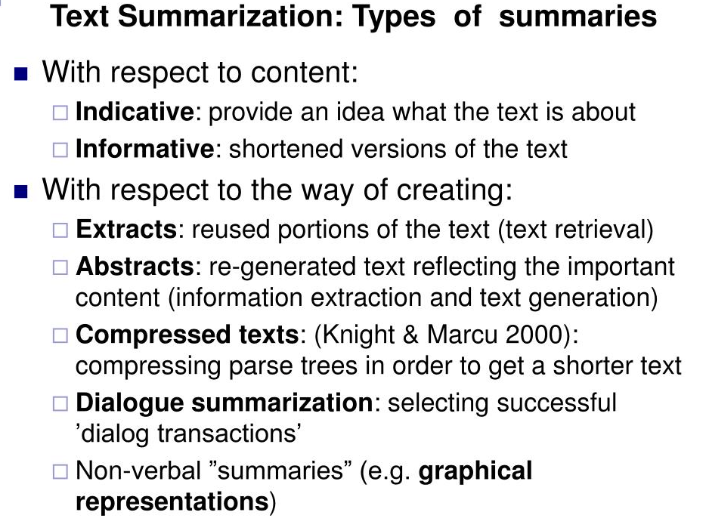
To develop an NLP-based model that summarizes the news articles and translates it to regional language.

**Theory :-**

Summarization is the task of condensing a piece of text to a shorter version that contains the main information from the original.

The need for quick acquisition and assimilation of useful insights from a large corpus of information on the Internet has driven the development of various automated summarization systems. These systems provide filtered and high-quality concise content to the users allowing them to work at unprecedented scale and speed.

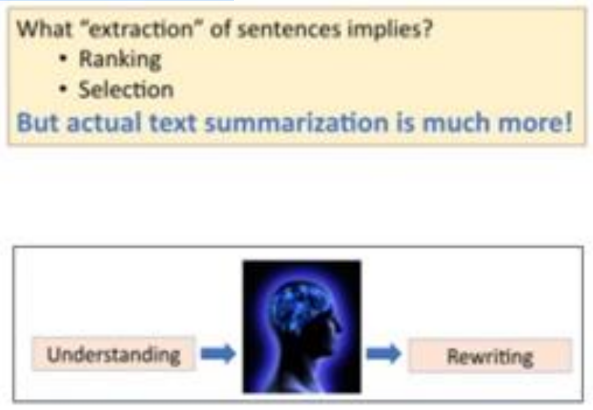




**Extraction**: - In Extractive text summarization , summary is generated by selecting a set of words, phrases, paragraphs or sentences from the original document.

➢Extract certain sentences

➢No issues with grammar



**Abstraction:** - Abstractive methods are based on semantic representation and then use natural language processing techniques to generate a summary that is nearer to summary generated manually. This kind of summary may contain words that are not found in the original document. Currently research is going on this method and demand for this method is more.

➢Produce “abstracts”

➢Content Understanding

➢generation

**Evaluative Summarization:** It aims to capture the opinion or the views of the author on a given topic/subject/product. Sometimes, it is also referred as review or opinion based summarization

**Indicative Summarization:** It portrays the key topics in the text reducing the length of the original text by 90%, includes metadata like writing style, length of a document, however fails to provide factual information. It helps to decide whether a user wants to read the document or not.

**Informative Summarization:** It contains content which is generally longer, reducing the length of original text by 70-80%. It includes facts and information which can

replace the original text.

**Semantic based abstractive summarization:**

The information item based method first identifies informative items by performing syntactic analysis on the text. Sentences are generated from these items by following the subject-verb-object structure using a sentence generator. The generated sentences are then ranked based on the average Document Frequency score. From this list, the highly ranked sentences are taken to create the summary. Finally, the semantic graph approach consists of three phases:

(a) representing the entire document by a Rich Se-

mantic Graph (RSG),

(b) applying heuristic rules to reduce the complexity of the semantic graph

(c) generation of abstract summary from the reduced graph.

**Challenges:**

Complexity: Summarization is one of the hardest problems of Natural Language Processing (NLP) as understanding text is very complex. It requires

understanding of semantics, inferential interpretations and discourse.

Word sense ambiguity: is ambiguity created as when an abbreviation has more than one acronym. In such cases, the acronym should be matched depending on the subject for better understanding.

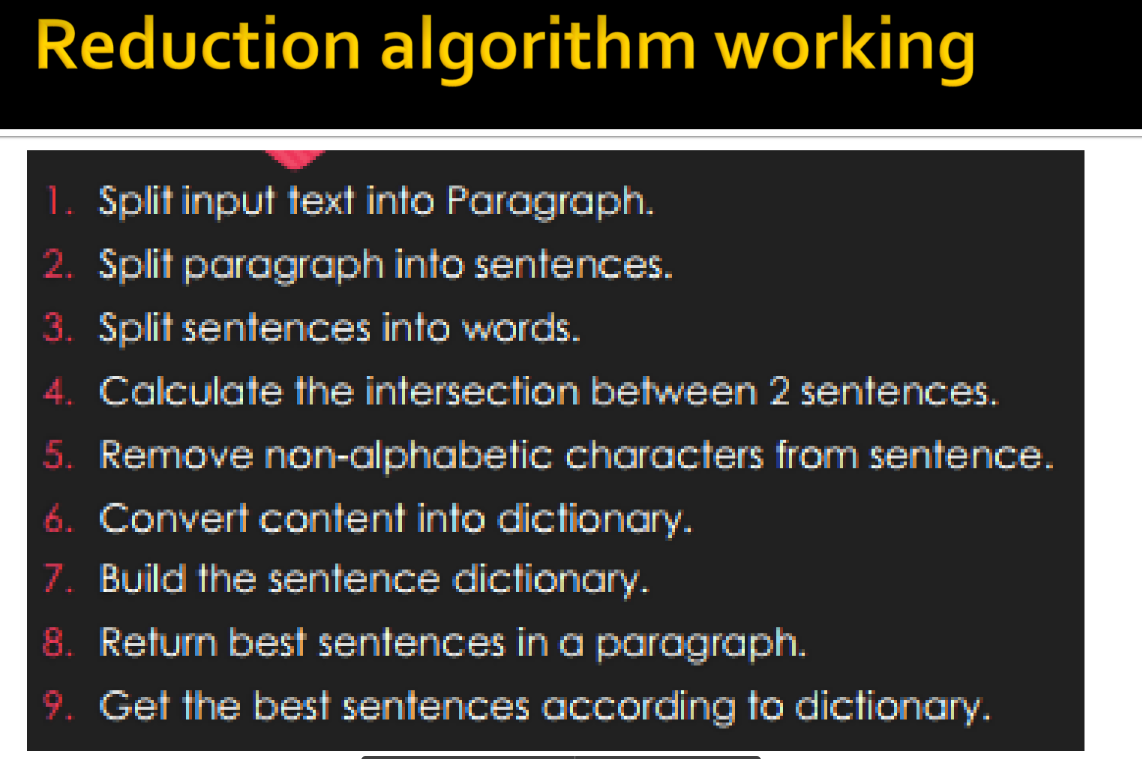
Summary evaluation: it is a hard problem. The main hurdle is building a fair gold standard against which the system generated results can be compared. It’s also hard

to determine what a correct summary is as it is subjective. Further, the existing popular evaluation metric (ROUGE score) is not suitable for evaluation of abstract summaries as ROUGE is the measure of n-gram matches. While the abstract summaries can have different words which are not a part of the original source texts. Subjectivity of summary: summarization quality varies from person to person. one per- son’s interest in a body of text is different than another’s,resulting in variation in quality of a summary

High reduction rate: extracts created from a single document summarizer usually aim to have five to thirty percent of the source text’s length. However, in

In the case of multi- document summaries for hand-held devices the compression rates are much lower. Which poses a serious challenge as such a high reduction rate is very hard to attain without having expert background knowledge.

While the traditional approach are ruled based and rely mostly on manually compiled features, recent interests has shifted towards using artificial neural network approaches which do not rely on manually compiled features and are rules independent.



**Code :-**

from newspaper import Article

from googletrans import Translator

import tkinter as tk

from tkinter import \*

import nltk

#GUI Window

top = tk.Tk()

top.update()

top.geometry('680x500')

top.title('News Article Summary')

# Label for Top

label = Label(top, background='#CDCDCD', font=('calibri', 20, 'bold'))

art\_label = ('arial', 16, 'bold')

art\_cont = ('verdana',12)

# Variables used for both the function

article = ''

art\_title = ''

article\_sum = ''

art\_content= ''

#canvas and frame

my\_canvas = Canvas (top, borderwidth=0, background="#ffffff")

frame = Frame(my\_canvas, background="#ffffff")

frame.pack(fill=BOTH, expand=TRUE)

vsb = Scrollbar(top, orient="vertical", command=my\_canvas.yview)

my\_canvas.configure (yscrollcommand=vsb.set)

vsb.pack(side="right", fill="y")

my\_canvas.pack(side="right", fill="both", expand=True)

my\_canvas.create\_window((0,0), window=frame, anchor="nw")

my\_canvas.pack (fill="both", expand=True)

# scrollbar

def onFrameConfigure(canvas):

'''Reset the scroll region to encompass the inner frame'''

canvas.configure(scrollregion=canvas.bbox("all"))

frame.bind("<Configure>", lambda event, canvas=my\_canvas: onFrameConfigure (my\_canvas))

# Translating and summary function

def translate\_news():

global article

global art\_title

global article\_sum

global art\_content

# Article

article = Article (news\_url.get())

article.download()

article.parse()

article.nlp()

translator = Translator()

art\_title = translator.translate(article.title, dest=btn2.get())

article\_sum = translator.translate(article.summary, dest = btn2.get())

art\_content = translator.translate(article.text, dest = btn2.get())

# Printing title

title\_label = Label(frame, text="Article Title:", font=art\_label)

title\_label.pack(pady=3)

article\_title = Text(frame, height=2, width=65, wrap='word', font=art\_cont, borderwidth=2)

article\_title.insert('end', art\_title.text)

article\_title.config(state="disabled")

article\_title.pack(pady=1)

# Printing Summary

summary\_label = Label(frame, text="Article Summary:", font=art\_label)

summary\_label.pack(pady=3)

text\_box = Text(frame, height=8, width=50, wrap='word')

text\_box.insert('end', article\_sum.text)

text\_box.pack(fill=BOTH, expand=True)

# Printing Publisher Date

publish\_label = Label (frame, text="Article Publish Date:", font=art\_label)

publish\_label.pack(pady=3)

article\_publish = Text(frame, height=1, width=50, font=art\_cont, borderwidth=2)

article\_publish.insert('end', article.publish\_date)

article\_publish.config(state="disabled")

article\_publish.pack(pady=1)

# Printing Image top link

link\_label = Label(frame, text="Article Image link:",font=art\_label)

link\_label.pack(pady=3)

label\_image = Text(frame, font=art\_cont, height=2, width=65, borderwidth=2)

label\_image.insert('end', article.top\_image)

label\_image.config(state="disabled")

label\_image.pack(pady=1)

# Downloading article Function

def download\_article():

file1 = open("News\_summary\_File.txt", "w+", encoding="utf-8")

file1.write("Title:\n")

file1.write(art\_title.text)

file1.write("\n\nArticle Text:\n")

file1.write(art\_content.text)

file1.write("\n\nArticle Summary: \n")

file1.write(article\_sum.text)

file1.write("\n\nArticle Publish Date:\n")

file1.write(str(article.publish\_date))

file1.write("\n\nArticle Image link:\n")

file1.write(article.top\_image)

file1.close()

#Url Entry field

Label(frame, text="URL", font=('arial', 18, 'bold')).pack()

news\_url = Entry (frame, width=65, borderwidth=2, font=("verdana", 11),)

news\_url.configure(highlightbackground="red", highlightcolor="red")

news\_url.pack(pady=5)

#Language selection

my\_label = Label (frame, text="Select Language in which you want to translate", font=art\_label)

my\_label.pack(pady=5)

# Radio font

font\_radio = ('calibri', 13)

btn2 = StringVar(value="en")

Radiobutton(frame, text="English", value="en", variable=btn2, font=font\_radio).pack(pady=5)

Radiobutton(frame, text="Marathi", value="mr", variable=btn2, font=font\_radio).pack(pady=5)

Radiobutton(frame, text="Hindi", value="hi", variable=btn2, font=font\_radio).pack(pady=5)

# Translation and summarization button

translate\_button = Button (frame, text="Translate and summarize",font=('verdana', 11, 'bold'), command=translate\_news).pack(pady=6)

# Downloading article button

download\_button = Button (frame, text="Download article", font=("verdana", 11, 'bold'),command=download\_article)

download\_button.pack(pady=6)

top.mainloop()

**Process :-**

**Calculate TF-IDF Vectors, Convert the preprocessed documents into TF-IDF vectors, which will be used to measure similarity.**

**Now, you can calculate the text similarity using the Jaccard similarity coefficient or any other similarity measure you prefer. Here, we'll use cosine similarity:**

**You can test the similarity between any two documents using the cosine\_similarities matrix:**

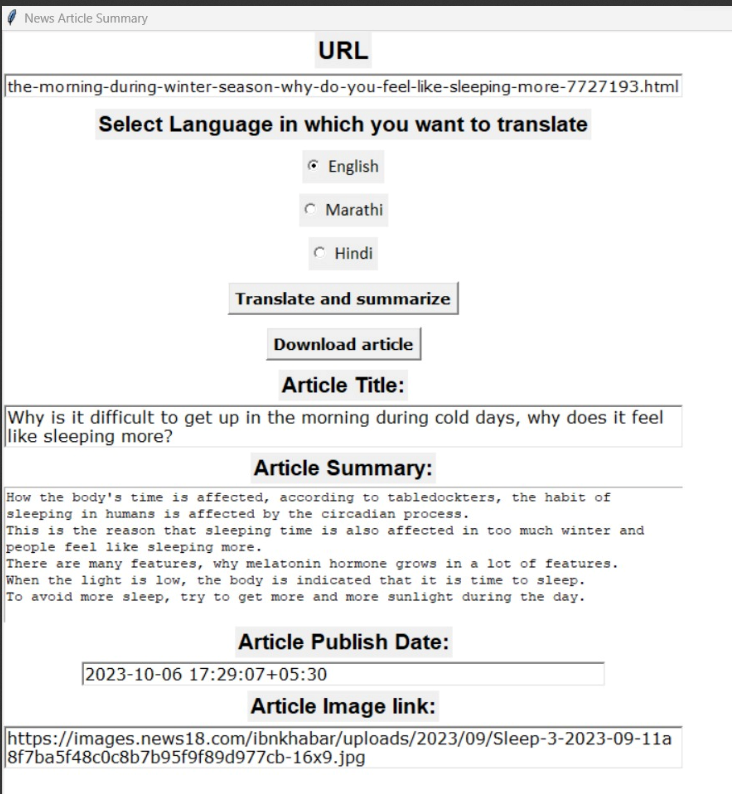
**Repeat the similarity test for different pairs of documents as needed for your project.**

**Input :-**

You can put any html link of any news article, For example

<https://hindi.news18.com/news/knowledge/why-is-it-difficult-to-wake-up-in-the-morning-during-winter-season-why-do-you-feel-like-sleeping-more-7727193.html>

**Output :-**

****