International Journal of Innovative Research in Engineering
Volume 4, Issue 2 (March-April 2023), PP: 476-480.
www.theijire.com **ISSN No: 2582-8746**

Book Summarization using NLP

Vrushali Agate[1], Shilotej Mirajkar[2], Gauri Toradmal[3]
[1,2,3]E&TC, Pune Institute of Computer Technology, Pune, Maharashtra, India.

How to cite this paper:
Vrushali Agate[1], Shilotej Mirajkar[2], Gauri Toradmal[3], "Book Summarization using NLP", IJIRE-V4I02-476-480.

https://www.doi.org/10.59256/ijire.2023040218

Abstract: *It is crucial to offer an enhanced system for swiftly and effectively extracting information in this modern era where the Internet is a wealth of knowledge. It is quite challenging for humans to manually extract the summary from a lengthy written document. On the Internet, there is a wealth of textual content. As a result, finding relevant papers from the many that are available and learning useful information from them is a challenge. An automatic text summary is crucial for resolving the two issues. The technique of extracting the most significant information from a document or group of related texts and condensing it into a concise version while maintaining its overall meaning is known as text summarizing. Text summarization reduces the length of the original text while maintaining the information and overall meaning. Large texts are particularly challenging for humans to manually summarize. There are two types of text summarizing techniques: extractive and abstractive. Selecting significant sentences, paragraphs, etc. from the original content and concatenating them into a shorter version constitutes an extractive summary method. Sentences' linguistic and statistical characteristics are used to determine their importance. Understanding the original material and retelling it in fewer words are the components of an abstractive summary method. It analyzes and interprets the content using linguistic techniques before identifying fresh ideas and expressions to create a new, concise copy that effectively communicates.*

Key Word: *NLP(Natural Language Processing),LDA(Latent Dirichlet Allocation).*

I.INTRODUCTION

1.1. Background

To extract the summary from lengthy texts is the goal of text summarizing. Utilizing a summary has the tremendous benefit of cutting down on reading time, effort, and expense. The approach used to summarize this material entails locating the key information that can be found in numerous documents. There are two methods to approach this: extractive and abstractive. You cancreate an extractive summary by picking the key phrases from the source text. By being aware of the key ideas in the document andreinterpreting them, one can create an abstractive summary. Machine learning is also used to analyze customer evaluations on any e-commerce site, such as Amazon. This information is useful for identifying user trends and habits.

1.2. Relevance

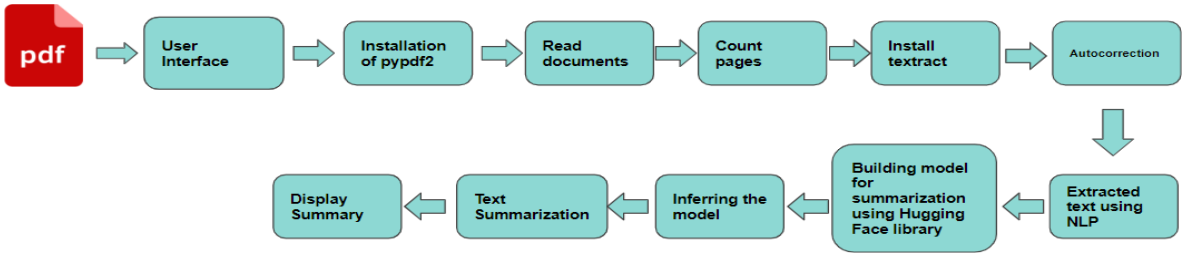
Automatic Text Summarization is a method that reduces lengthy texts and creates summaries to convey the intendedcontent. It is a common issue in machine learning and natural language processing (NLP). Text summarizing entails outlining the main ideas of the text in order to provide a concise, coherent, and fluid summary of a lengthy text document. Some of the main difficulties encountered in the process of text summarization include text detection, interpretation, summary generation, and examination of the resulting summary. Finding important terms in the text and exploiting them to find pertinent information to add in the summary are crucial jobs in extraction-based summarization. The following are the two methods that are utilized for text summarization: 1) Extractive 2) Abstractive

II.MATERIALS AND METHODS

2.1. Technical Approach:

The user will upload a pdf of the book on the user interface which is a website. The next step is to extract text from the uploaded pdf. At the backend, using various python libraries and functions to extract the text from pdf. Then the next and important step is an abstractive summarization using NLP. Abstractive summarization is the new state of art method, which generates new sentences that could best represent the whole text. This is better than extractive methods where sentences are justselected from original text for the summary. A simple and effective way to implement abstractive summarization is through the Huggingface's transformers library. Hugging Face supports state of the art models to implement tasks such as summarization, classification, etc.. Some common models are GPT-2, GPT-3, BERT, Open AI, GPT, T5.Summary will be displayed on the website.

Block Diagram



2.2. Algorithm

A. Text Extraction Algorithm :

1. import libraries
2. Using PyPDF2, read our document: Use the Pdf File Reader() method from PyPDF2, a library for extracting document information like tables and pages, to read our manifesto in pdf format. documents that were divided into pages. Page by page, combine them. should use the num Pages() function to get the number of pages in our document and the get Document Info() function to obtain the document's details. There are several beneficial features that can be used to check other items.
3. The title of the document, the operating system that was used to type it, and the dates that it was generated and edited were allobtained from the outputs of our two previous procedures. The total number of pages in our document was also provided to us
4. Printed the texts that were extracted from the pdf file.
5. Extracted the texts from the pdf file and print it-:We will use a textract package to extract our texts from the document .Loop through all the pages in the document and extract the text from it. The if statement check if our document returned words from theloop above using the extract Text() function. This is done since PyPDF2 cannot read scanned documents. If the above returns a false, then run the Optical Character Recognition (OCR) textract to convert scanned/image based Pdf files to text.
6. printed out our texts to see what it contains which was converted to lowercase using the lower() function
7. Auto corrected and we produced the extraction of text.

B. Text Summarization Algorithm

1. Installed transformer and imported pipeline.
2. Transformers: is that it provides Pre Trained models with weights that can be easily instantiated through from a pre trained model.
3. Each task is associated with pipeline (), it is simpler to use the general pipeline() abstraction which contains all the task-specific pipelines. The pipeline () automatically loads a default model and a preprocessing class capable of inference for task.
4. Generated chunks of the sentences the purpose to do so is to increase the speed and reduce the run time.
5. Created an os environment and summarized the text result.

III.RESULT

```
!pip install PyPDF2
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.8/dist-packages (3.0.1)
Requirement already satisfied: typing_extensions>=3.10.0.0 in /usr/local/lib/python3.8/dist-packages (from PyPDF2) (4.4.0)
```

```
import PyPDF2
import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)


filename = 'thepresent.pdf'
open_filename = open(filename, 'rb')

ind_manifesto = PyPDF2.PdfReader(open_filename)


ind_manifesto.metadata
```

```
{'/Title': 'Slide 1',
 '/Author': 'sompong.y',
 '/CreationDate': "D:20130406024326-07'00'",
 '/ModDate': "D:20130406024326-07'00'",
 '/Producer': 'Microsoft® PowerPoint® 2010',
 '/Creator': 'Microsoft® PowerPoint® 2010'}
```

```
total_pages = len(ind_manifesto.pages)
total_pages
```

```
34
```

```
!pip install textract
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: textract in /usr/local/lib/python3.8/dist-packages (1.6.5)
Requirement already satisfied: six~=1.12.0 in /usr/local/lib/python3.8/dist-packages (from textract) (1.12.0)
Requirement already satisfied: extract-msg<=0.29.* in /usr/local/lib/python3.8/dist-packages (from textract) (0.28.7)
Requirement already satisfied: pdfminer.six==20191110 in /usr/local/lib/python3.8/dist-packages (from textract) (20191110)
Requirement already satisfied: argcomplete~=1.10.0 in /usr/local/lib/python3.8/dist-packages (from textract) (1.10.3)
Requirement already satisfied: beautifulsoup4~=4.8.0 in /usr/local/lib/python3.8/dist-packages (from textract) (4.8.2)
Requirement already satisfied: docx2txt~=0.8 in /usr/local/lib/python3.8/dist-packages (from textract) (0.8)
Requirement already satisfied: xlrd~=1.2.0 in /usr/local/lib/python3.8/dist-packages (from textract) (1.2.0)
Requirement already satisfied: SpeechRecognition~=3.8.1 in /usr/local/lib/python3.8/dist-packages (from textract) (3.8.1)
Requirement already satisfied: chardet==3.* in /usr/local/lib/python3.8/dist-packages (from textract) (3.0.4)
Requirement already satisfied: python-pptx~=0.6.18 in /usr/local/lib/python3.8/dist-packages (from textract) (0.6.21)
Requirement already satisfied: sortedcontainers in /usr/local/lib/python3.8/dist-packages (from pdfminer.six==20191110->textract) (
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.8/dist-packages (from pdfminer.six==20191110->textract) (3.1
Requirement already satisfied: soupsieve>=1.2 in /usr/local/lib/python3.8/dist-packages (from beautifulsoup4~=4.8.0->textract) (2.
Requirement already satisfied: compressed-rtf>=1.0.6 in /usr/local/lib/python3.8/dist-packages (from extract-msg<=0.29.*->textract)
Requirement already satisfied: olefile>=0.46 in /usr/local/lib/python3.8/dist-packages (from extract-msg<=0.29.*->textract) (0.46)
Requirement already satisfied: imapclient==2.1.0 in /usr/local/lib/python3.8/dist-packages (from extract-msg<=0.29.*->textract) (2
Requirement already satisfied: ebcdic>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from extract-msg<=0.29.*->textract) (1.1.1)
Requirement already satisfied: tzlocal>=2.1 in /usr/local/lib/python3.8/dist-packages (from extract-msg<=0.29.*->textract) (4.2)
Requirement already satisfied: Pillow>=3.3.2 in /usr/local/lib/python3.8/dist-packages (from python-pptx~=0.6.18->textract) (7.1.2)
Requirement already satisfied: XlsxWriter>=0.5.7 in /usr/local/lib/python3.8/dist-packages (from python-pptx~=0.6.18->textract) (3
Requirement already satisfied: lxml>=3.1.0 in /usr/local/lib/python3.8/dist-packages (from python-pptx~=0.6.18->textract) (4.9.2)
Requirement already satisfied: backports.zoneinfo in /usr/local/lib/python3.8/dist-packages (from tzlocal>=2.1->extract-msg<=0.29.*
Requirement already satisfied: pytz-deprecation-shim in /usr/local/lib/python3.8/dist-packages (from tzlocal>=2.1->extract-msg<=0.2
Requirement already satisfied: tzdata in /usr/local/lib/python3.8/dist-packages (from pytz-deprecation-shim->tzlocal>=2.1->extract-
```

```
import textract
```

```
count = 0
text  = ''

# Lets loop through, to read each page from the pdf file
while(count < total_pages):
    # Get the specified number of pages in the document
    mani_page  = ind_manifesto.pages[count]
    # Process the next page
    count += 1
    # Extract the text from the page
    text += mani_page.extract_text()


if text != '':
    text = text


else:
```

```
!pip install autocorrect
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: autocorrect in /usr/local/lib/python3.8/dist-packages (2.6.1)
```

```
!pip install nltk
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.8/dist-packages (3.7)
Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (from nltk) (7.1.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from nltk) (4.64.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.8/dist-packages (from nltk) (2022.6.2)
```

```
import nltk.tokenize
```

```
from nltk.tokenize import word_tokenize
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
from autocorrect import Speller
```

```
def to_lower(text):

    """
    Converting text to lower case as in, converting "Hello" to  "hello" or "HELLO" to "hello".
    """

    # Specll check the words
    spell  = Speller(lang='en')

    texts = spell(text)

    return ' '.join([w.lower() for w in word_tokenize(text)])

lower_case = to_lower(text)
print(lower_case)
```

```
the present . “ living in the present , learning from the past and planning for the future ” thank you very much sompong yusoontorn
```

```
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: transformers in /usr/local/lib/python3.8/dist-packages (4.26.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (from transformers) (6.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in /usr/local/lib/python3.8/dist-packages (from transformers) (0.12.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from transformers) (3.9.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (1.21.6)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.8/dist-packages (from transformers) (4.64.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (2022.6.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from transformers) (23.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.8/dist-packages (from transformers) (0.1
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from transformers) (2.25.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.8/dist-packages (from huggingface-hub<1.0,>=0.1
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (1.24
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (2022.12
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (3.0.4)
```

```
from transformers import pipeline


def generate_chunks(inp_str):
    max_chunk = 500
    inp_str = inp_str.replace('.', '.<eos>')
    inp_str = inp_str.replace('?', '?<eos>')
    inp_str = inp_str.replace('!', '!<eos>')

    sentences = inp_str.split('<eos>')
    current_chunk = 0
    chunks = []

    for sentence in sentences:
        if len(chunks) == current_chunk + 1:
            if len(chunks[current_chunk]) + len(sentence.split(' ')) <= max_chunk:
                chunks[current_chunk].extend(sentence.split(' '))
            else:
                current_chunk += 1
                chunks.append(sentence.split(' '))
        else:
            chunks.append(sentence.split(' '))

    for chunk_id in range(len(chunks)):
        chunks[chunk_id] = ' '.join(chunks[chunk_id])
    return chunks

import os


os.environ["CUDA_VISIBLE_DEVICES"] = "0"


summarizer = pipeline("summarization", model="t5-base", tokenizer="t5-base", framework="tf")
```

```
All model checkpoint layers were used when initializing TFT5ForConditionalGeneration.

All the layers of TFT5ForConditionalGeneration were initialized from the model checkpoint at t5-base.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFT5ForConditionalGeneration fo
/usr/local/lib/python3.8/dist-packages/transformers/models/t5/tokenization_t5_fast.py:155: FutureWarning: This tokenizer was incorr
For now, this behavior is kept to avoid breaking backwards compatibility when padding/encoding with `truncation is True`.
- Be aware that you SHOULD NOT rely on t5-base automatically truncating your input to 512 when padding/encoding.
- If you want to encode/pad to sequences longer than 512 you can either instantiate this tokenizer with `model_max_length` or pass
- To avoid this warning, please instantiate this tokenizer with `model_max_length` set to your preferred value.
  warnings.warn(
```

```
max = 150
min = 50
sentence = lower_case
chunks = generate_chunks(sentence)
res = summarizer(chunks,
                         max_length=max,
                         min_length=min
                         )
text = ' '.join([summ['summary_text'] for summ in res])
        # st.write(result[0]['summary_text'])
print(text)
```

```
Token indices sequence length is longer than the specified maximum sequence length for this model (575 > 512). Running this sequenc
/usr/local/lib/python3.8/dist-packages/transformers/generation/tf_utils.py:603: UserWarning: You have modified the pretrained model
  warnings.warn(
spencer johnson , m. d. is an international best -selling author whose books help millions of people discover simple truths they ca
```

IV.DISCUSSION

From our summarization result we have found that by reducing all sentences that do not contain any geographical information may lead to a loss of information, since there may exist links between that reduced sentences. Therefore, we will analyzethis issue in detail, studying graph based algorithms that capture the relationship between sentences. Coverage of All

Important Information Is Assured The human eye can overlook significant details, while computer software does not. Any piece of material should allow readers to pick out the information that will be most helpful to them. Finding all the important information in a document is made simple for the user by the automatic text summarizing technique. Modern summary methods are all extractive innature, however abstractive summarization is becoming embraced by the community. Although a comprehensive comprehension and processing of natural language would be necessary for a complete abstractive summarisation, a hybrid or shallow abstractive summarization can be accomplished using sentence compression and textual entailment techniques. Textual entailment aids in identifying condensed versions of longer texts that convey the same content. Textual entailment enables us to create summariesthat are shorter and more to the point. The method that was put into practise for this thesis can serve as a framework for the research community to comprehend and broaden the applicability of cognitive and symbolic approaches in diverse businessdemand domains. The diversity and depth of information are still being increased through summarizing research, and it works to provide users with responses that are focused and cohesive.

V.CONCLUSION

One of the biggest issues in the field of natural language processing is text summarization. Methods such as Sentence Extraction, Paragraph Extraction, Machine Learning, Deep Understanding, and even others that combine all of these techniques with Classic NLP Techniques (Semantic Analysis, etc.). In light of these successes, there is still a tonne of research to be done in the field of text summarization since a meaningful summary is still challenging to produce across all disciplines and languages. The internet is expanding quite quickly over time, and along with it, data and information are growing as well. Large amounts of data will be challenging for humans to summarize. Due to the vast amount of data, automatic text summarizing is required. Text summarization is an intriguing area of machine learning that is attracting more study. We can anticipate developments that will aid in properly and fluently condensing lengthy text texts as research in this field progresses. Here, we can claim that text summarizing using NLP was efficiently done in accordance with the problem specification. With the help of the text summaries in this project, we were able to fix the issue. We did our best to make these summaries as significant in terms of text intention as feasible. We can add many capabilities to our online apps, such as the ability to accept text input in pdf format by uploading it directly to our input box for text summarizing.

References

1. *Andhale, N., & Bewoor, L. A. (2016). An overview of Text Summarization techniques. 2016 International Conference on Computing Communication Controland Automation (ICCUBEA). doi:10.1109/iccubea.2016.7860024*
2. *Boorugu, R., & Ramesh, G. (2020). A Survey on NLP based Text Summarization for Summarizing Product Reviews. 2020 Second International Conferenceon Inventive Research in Computing Applications (ICIRCA). doi:10.1109/icirca48905.2020.9183*
3. *Neelima Bhatia, ArunimaJaiswal, "Automatic Text Summarization: Single and Multiple Summarizations ", International Journal of Computer Applications.*
4. *Bui, D. D. A., Del Fiol, G., & Jonnalagadda, S. (2016). PDF text classification to leverage information extraction from publication reports. Journal ofBiomedical Informatics, 61, 141–148. doi:10.1016/j.jbi.2016.03.026*
5. *Dipanjan Das, Andre F.T. Martins, "A Survey on Automatic Text Summarization",Language Technologies Institute, Carnegie Mellon University, November2007.*
6. *Hamid Jelodar, Yongli Wang, Chi Yuan , "Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey" Nanjing University ofScience and Technology , November 2007*
7. *NLTK Documentation: https://www.nltk.org/*
8. *HuggingFace: https://huggingface.co/docs/hub/models-libraries*