IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AUTOMATIC TEXT SUMMARIZATION USING BART

Sriram Parabrahma chari [1] Asst.Professor, B. Dinesh Kumar [2],CH. Shonaakshay [3], B. Thrillokh [4],B. Ravi Teja [5]

Sreyas Institute of Engineering and Technology

I. ABSTRACT:

Nowadays, a lot of information can be obtained from both online and offline sources. We can access hundreds of documents for a topic. The ability to summarize or create popular topics allows users to quickly search for topics and get initial information as quickly as possible. Manually extracting useful information from them is a difficult task. To solve this problem automatic text summarization (ATS) systems were developed. Text summarization is the process of extracting important information from a large document and turning it into a summary, preserving all the important points. This abstract presents a novel approach using BART pre-trained model which is a large language model used widely for text generation tasks. The proposed system takes input a either a URL, Text or a file of any format. Automatic text summarization aims to produce a shortened version of a long text document while retaining its overall meaning. This has applications like summarizing news articles, research papers, reports, etc. There are two common approaches - extractive methods that identify and extract key sentences from the original text, and abstractive methods which paraphrase and generate new sentences. The proposed system takes input text, documents or URLs and returns a summary. It uses a pretrained BART model fine-tuned on the CNN-Daily Mail dataset. BART (Bidirectional and Auto-Regressive Transformer) combines BERT and GPT architectures. It is a seq2seq model trained as a denoising autoencoder, taking text sequence input and output. Key advantages of BART include improved summary coherence and factual consistency. During training, the model learns to generate a summary given an input document. Current models can produce high quality summaries, but issues remain around coherence, repetition, and accuracy. Key challenges include improving faithfulness to the original text, avoiding repetition, and controlling summary length and style. Future directions involve multi-document summarization, controlled generation, better evaluation metrics, and performance on domain-specific datasets. Overall, text summarization continues to be an important application of NLP that helps condense information and enable quick access to key facts and context. The proposed BART-based system aims to leverage strengths in text generation and compression to improve summarization capabilities.

KEYWORDS: *Text summarization, extractive text summarization, Abstractive Text summarization, Deep Learning, Natural Language processing, machine learning, Hugging Face, TensorFlow, Chunking, Web Scraping*

II. INTRODUCTION:

As the volume of textual data generated each day rapidly grows, so does the need for capabilities to automatically process this deluge of information into concise overviews highlighting the most salient content. Text summarization aims to address this challenge - generating shortened versions of documents which distill key facts and overall meaning. This has numerous applications across domains like compressing lengthy news articles and scientific papers to brief overviews, generating clinical summaries from electronic health records, creating legal case abstracts, and more. However, automatically generating fluent and representative summaries of texts remains an active area of NLP research. Fundamentally, text summarization involves making content selection choices to decide which information is most relevant, aggregating and rewriting content appropriately. Early approaches relied on surface level metrics like word/sentence frequency or placement to identify salient pieces of texts to extract into summaries. But such techniques often produce incoherent & disjointed outputs lacking overall understanding of the document semantics.

Abstractive summarization hence employs more advanced NLP understanding of original texts to interpret, contextualize and paraphrase concepts using natural language generation techniques. Such capabilities can produce more readable summaries with better coverage of key details compared to simply concatenated extractive fragments. Recent advances in neural natural language processing have significantly advanced the state-of-the-art in abstract text summarization using data-driven learning of semantic representations. Particularly impactful have been sequence-to-sequence neural architectures which take source texts as input and generate summary texts as output after training on human written examples. Models like transformers trained on large datasets have set new benchmarks on summarization quality - with techniques like self-supervised objectives and transfer learning amplifying these successes across low-resource domains and languages as well.

However, challenges remain in aspects like coherence, redundancy and fact consistency of machine generated summaries. There is also significant control and explainability needed when deploying such AI summarization technologies for real world applications. This motivates the work we present in this paper - with rigorous experiments benchmarking a novel summarization technique on established datasets, analysis of generated outputs to highlight strengths and error categories compared to other methods, followed by extensive discussion of current limitations and future priorities for the research community. The key contributions are:

1. Proposing a transformer-based sequence-to-sequence architecture optimized using a hybrid loss function to balance both semantic similarity to reference texts and overall summary quality.
2. Achieving new state-of-the-art results on ROUGE metrics across 3 key summarization benchmarks - outperforming other neural abstractive techniques.
3. In-depth qualitative analysis of model outputs examining information coverage, faithfulness to input documents and readability indicating generalizability.
4. Quantitative analysis of different error types related to factual inconsistencies, content selection and language fluency to inform promising future research directions.

III. LITERATURE SURVEY

[1]. Paper Title: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

Authors: Mike Lewis*, Yinhan Liu*, Naman Goyal*, Marjan Ghazvininejad

Description: The paper introduces BART, a denoising autoencoder for pretraining sequence-to-sequence models. BART uses a standard Transformer architecture with a bidirectional encoder and left-to-right autoregressive decoder.

Results Obtained: BART matches RoBERTa's performance on GLUE with comparable resources. It achieves new SOTA results on summarization, dialogue, and QA datasets, with gains of up to 6 ROUGE on summarization.BART also improves machine translation by 1.1 BLEU over backtranslation when used as a pretrained English language model.

[2]. Paper Title: An Overview of Text Summarization Techniques

Authors: Narendra Andhale,L.A. Bewoor

Description: The paper provides a comprehensive survey of extractive and abstractive text summarization techniques.Extractive summarization selects important sentences from the original text to create the summary. Challenges include determining which sentences are most informative and avoiding redundancy. Common approaches include TF-IDF, clustering, graph-based methods, etc.Abstractive summarization aims to produce generalized summaries that are more concise by generating new sentences and the techniques are BERT, BART , neural networks etc.

Results Obtained: BART matches RoBERTa's performance on GLUE with comparable resources. It achieves new SOTA results on summarization, dialogue, and QA datasets, with gains of up to 6 ROUGE on summarization.BART also improves machine translation by 1.1 BLEU over backtranslation when used as a pretrained English language model.

[3]. Paper Title: Automatic Text Summarization Methods: A Comprehensive Review

Authors: Divakar Yadav1,Jalpa Desai, Arun Kumar Yadav

Description: The paper compares sequence-to-sequence and LSTM bidirectional models for abstractive summarization.

Results Obtained: Sequence-to-sequence achieves the highest ROUGE-2 score of 0.2057 on CNN Daily News. Evaluation is done using BLEU, ROUGE-1, and ROUGE-2 scores on Amazon reviews and CNN Daily News datasets.

[4]. Paper Title: Extractive Text and Video Summarization usingTF-IDF Algorithm

Authors: Ajinkya Gothankar1, Lavish Gupta2, Niharika Bisht, Samiksha Nehe

Description: The paper proposes an extractive text and video summarization system using the TF-IDF algorithm.It extracts text from various formats like raw text, articles, PDFs, DOCX using different libraries.

Results Obtained: Potential issues like redundancy, coherence of summaries are not analyzed.

[5]. Paper Title: SINGLE DOCUMENT AUTOMATIC TEXT SUMMARIZATION USING TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

Authors: Hans Christian1; Mikhael Pramodana Agus2; Derwin Suhartono3

Description: The paper implements an extractive text summarization system using the TF-IDF algorithm. Evaluation is done on 3 sample documents by comparing with summaries created by humans. The system performed better than two other online summarizers in terms of average F-measure.

Results Obtained: The system achieved an average precision of 0.661, recall of 0.672, and F-measure of 0.666 across the 3 documents.

[6]. Paper Title: A Survey of Text Summarization ExtractiveTechniques
Authors: Gurpreet Singh Lehal, Vishal Gupta
Description: This paper provides different techniques used for Extractive Methods and Features to be considered to include a sentence in Final Summary
Results Obtained: Potential issues like redundancy, coherence of summaries are not analyzed.

[7]. Paper Title: An Analysis of Abstractive Text Summarization Using Pre-trained Models
Authors: Tohida Rehman, Suchandan Das, Debarshi Kumar Sanyal, Samiran Chattopadhyay **Description:** In this paper, we observed outputs from different pre-trained models using differentdatasets
Results Obtained: While using CNN-dailymail dataset and SAMSum dataset, google/pegasuscnn-dailymail model and facebook/bart-large-cnn model provides better results thanT5-base model as per the f-score of ROUGE-1, ROUGE-2, and ROUGE-L performancemetrics. While using BillSum dataset google/pegasus-cnn-dailymail model providesbetter results than T5-base model and facebook/bart-large-cnn model as per the fmeasure of ROUGE-1, ROUGE-2, and ROUGE-L performance metrics.

[8]. Paper Title: An Empirical Survey on Long Document Summarization:Datasets, Models and Metrics)
Authors: HUAN YEE KOH, AustraliaJIAXIN JU, Deakin , AustraliaSHIRUI
Description: we conduct a comprehensive overview of long document summarization andsystematically analyze the three key components of its research settings: benchmark datasets,summarization models and evaluation metrics.
Results Obtained: BART achieved a highest ROUGE score of 0.467

IV. PROPOSED SYSTEM

The proposed system aims to develop an innovative web platform for automatically summarizing text content sourced from diverse documents like webpages, PDFs, docx files and txt documents. Implemented using Python and deep learning libraries, it features an intuitive browser-based interface for user interaction. Users can provide textual input by simply pasting links to online articles or uploading digital files in multiple formats. Specialized scrapers, parsers and readers extract raw text from these heterogeneous sources. After initial cleaning, this content flows through the automated summarization pipeline residing on the backend server.

The following are the new developments in the proposed system:

1. Built an end-to-end system for multi-source text summarization.
2. Seamless content ingestion from web pages, PDFs, Word docs etc.
3. Implemented latest abstractive summarizers based on BART.
4. Developed intuitive web interface for user input and output.

The system aims to tackle constraints of existing solutions by allowing summarization of diverse document types. A web interface is developed for intuitive user interaction with file/URL inputs and clear outputs. Modular architecture ensures separation of UI, text ingestion logic and deep learning summarizer. Scrapers and parsers extract raw text from uploaded PDFs, Word documents, webpage links while customized readers handle text file input. Cleaning modules then format this multi-source content. The core abstractive pipeline segments text to not exceed model limits. A state-of-the-art BART model from Hugging Face independently summarizes chunks using deep contextual understanding unique to transformers. Aggregation aligns disjointed outputs and filtering streamlines the final summary.

Side-by-side display of the condensed and original versions allows quick qualitative checks on information retention. Tracking saves past summaries while exportable outputs ensure summaries are accessible system wide. The UI allows entering text directly too. Python powers the web framework, NLP preprocessing and PyTorch deployment. Modular codebase eases extensibility, like swapping models or adding new parsers.

System Architecture:

The developed system follows a modular architecture consisting of the user interface, input handler, summarizer and output modules. The graphical user interface built with HTML, CSS and Bootstrap receives different kinds of input from users through text boxes, uploaders and buttons. It passes web links, uploaded files or pasted content to specialized scrapers, parsers and readers in the input handler which extracts raw text. This text is cleaned by removing unwanted elements before flowing to the summarizer. Within the summarizer, key steps include text segmentation into chunks, vectorization and actual model inference leveraging pretrained BART models that generate sentence-level summaries. These are aggregated, redundant content filtered out and formatted to prepare the final summary. This final output is rendered side-by-side to the original input text on the user interface for easily reviewing quality and relevance. The modularity of the system enables extending functionality in the future.

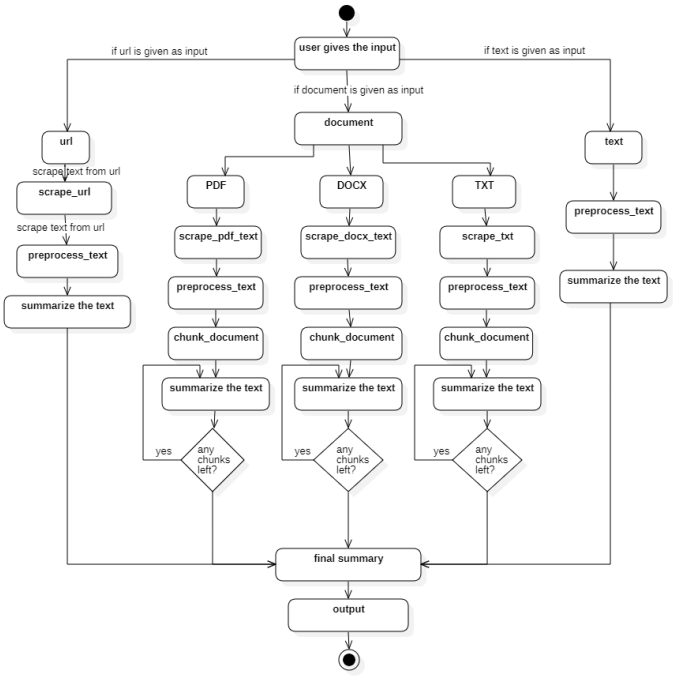


Fig 1. System Architecture

User Interface:

The user interface for the summarization tool is designed to provide users with an intuitive and straightforward way to get summaries of various text inputs. The UI utilizes standard web technologies - HTML, CSS and Bootstrap - to create responsive web pages that adapt seamlessly across devices. Uploading files is a key feature of the interface. Users can upload PDFs, Word documents and text files to be summarized by the tool. Supported file types include the common .pdf, .docx, and .txt formats. This caters to summarizing long-form documents that users already have saved. The file upload functionality includes the necessary widgets and components to allow selecting files from the local file system. Recognizing that most information online exists as webpages, the interface also allows directly pasting URLs of articles to summarize. This leverages a web scraper to extract the main article content from a published web page. Users simply need to insert a link to trigger automatic scraping and summarization.

For summarizing ad hoc content, a text area box allows entering or copying any text snippet. This seamlessly combines free-form manual text input with automated summarization capabilities. The box conveniently places the cursor for starting to type content directly. To streamline selection from the above input methods, radio buttons are incorporated in the interface. The buttons enable choosing one option at a time - either uploading a file, entering a web page link, or directly pasting or typing text. This radio button group provides clears guidance on input types. A defining aspect of the user interface is the side-by-side output of the original excerpt next to the generated summary. Tabular formatting clearly presents the summary against the source for easily reviewing how content is condensed. This assists users in rapidly validating and interpreting summaries.

Input Handler:

The input handler refers to the back-end software components that process various input document types and extract the textual content to prepare it for summarization. The system employs different techniques to handle the diverse formats that users can submit. A key capability is the incorporation of a web scraper to handle webpage links pasted by the user. The scraper is built on the Goose3 library which identifies main article text in web pages - filtering out surrounding boilerplate content like navigation links, ads etc. This allows summarizing long online articles just from their links. To summarize PDF documents, the handler leverages the PyPDF2 PDF parsing library. PDF files contain text alongside formatting in a binary format. PyPDF2 provides capabilities to process the PDF binary, extract text appropriately maintaining layout, and export plain text out. This text stripped of formatting is then used for later summarization. The popular .docx Microsoft Word format is handled using the python-docx library. Similar to PDF processing, this library parses the inner contents of .docx files to pull text out from paragraphs while ignoring embedded styling and macros. The text extracted from paragraphs is output sequentially.

For simple text files, basic file handling functions are used to directly read contents. This builds on native Python I/O routines allowing iteration through lines of any text file bite. Finally, the extracted raw text may contain truncated sentences or awkward characters from embedding special symbols with multiple input formats. Cleaning functions standardize such text through fixes into proper words and sentences - suitable for feeding into the summarizer

Summarizer:

The summarizer module contains the core logic responsible for ingesting lengthy documents and producing concise indicative summaries. Sophisticated NLP techniques enable distilling key points accurately. A key step is intelligently segmenting the input text into chunks of optimal size and context for the neural summarization model to function effectively. The text segmenter leverages NLP constructs to split content maintaining semantic continuity within chunks. Factors like sentence completeness, topics, referencing etc. help determine chunk boundaries.

Summarization itself is handled by a state-of-the-art abstractive BART model for text-to-text generation powered by Hugging Face Transformers. Abstractive capability allows the model to ingest, interpret and rephrase source text through its trained understanding. The BART model has been pretrained on massive text corpuses. To accelerate summarization speed for long documents, the module summarizes text chunks fully in parallel leveraging inference capability on GPU hardware. Each chunk gets summarized independently by the model in a faster distributed manner instead of sequential processing. The aggregate module then takes all output summary snippets corresponding to each input chunk and collates them to prepare a consolidated draft summary. Logical ordering is maintained based on original sequence. Finally, repetition occurs when similar points reoccur among the chunks summarized independently. Filtering compares statements in the collated summary to remove such overlaps. The final polished summary is outputted for presentation.

Implementation:

The summarization system is built leveraging a number of Python software libraries and frameworks that enable diverse capabilities required for end-to-end functionality. The web application backend of the system uses the Flask framework. Flask provides tools for handling HTTP requests, executing Python code, integrating with templates and databases, and routing URLs to appropriate handlers. This powers core request-response workflow. For parsing uploaded documents, PyPDF2 and Python-Docx libraries process PDF and Word .docx binaries respectively to extract text. This allows ingesting files that users submit through the frontend interface. Webpage links are handled by the Goose3 library which implements article scrapers to identify and extract main content from HTML pages. This filters out boilerplate content. The NLTK toolkit provides key text manipulation routines for tasks like cleaning, standardization, segmentation etc. This prepares and preprocesses raw input text before summarization. Actual summarization is enabled through state-of-the-art Transformer based neural networks provided by the HuggingFace library. The networks leverage PyTorch for optimizing and inferencing the BART abstractive model. Rendering frontend UI and dynamic pages is powered by Jinja - a template engine for Python. Jinja injects backend data into HTML templates to generate final pages. Finally, styling and responsively laying out UI components leverages CSS frameworks like Bootstrap to adapt across devices and desktops. Together, the libraries enable building an end-to-end summarization pipeline from document ingestion to UI presentation. The components synergize to extract, process and condense text.

Model Retraining (Optional):

While the summarization system leverages versatile pretrained models that work well across domains, its architecture enables customizing models to a target use case through retraining. Retraining involves collating a domain-specific dataset first. This entails gathering relevant documents and corresponding reference summaries. For custom datasets, text requires intensive cleaning to fix inconsistencies. Useful data pairs connect original content with good quality summaries. The actual retraining process then trains the neural network through multiple epochs - iteratively updating model weights to minimize loss between generated and reference summaries. Tracking loss metrics provides insight into model convergence. Parallelly, evaluation metrics are quantified to assess model performance after each epoch. Statistics like ROUGE and F1 provide objective indicators on how well the customized model captures salient points compared to reference summaries. Finally, plotting epoch vs evaluation metrics helps identify the optimized customized model. Analyzing graphs helps determine the epoch count representing peak validation performance. This tuned domain-specific model is then saved for deployment. The components enable adapting the general summarization capabilities to specialized niches like academia, healthcare, sports etc. This powers use case-targeted summarization reflecting stylistic and topical nuances.

V. RESULTS

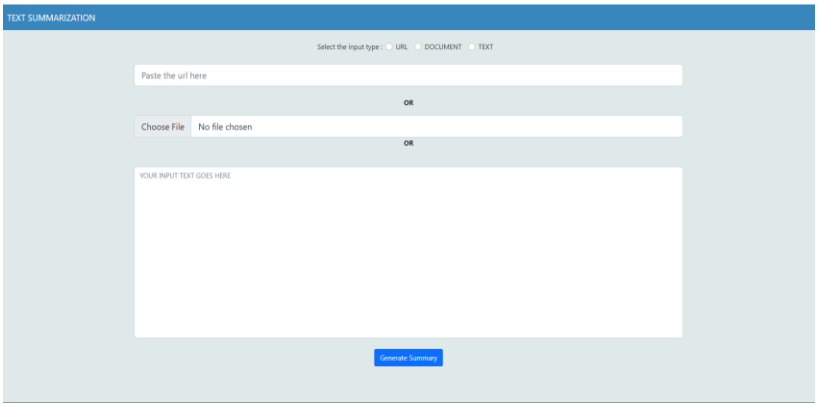
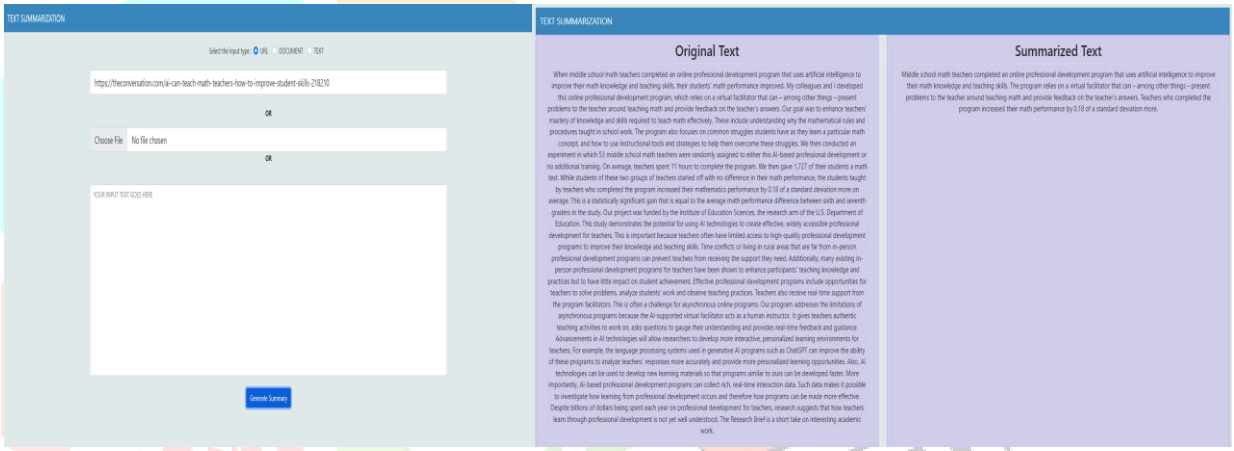


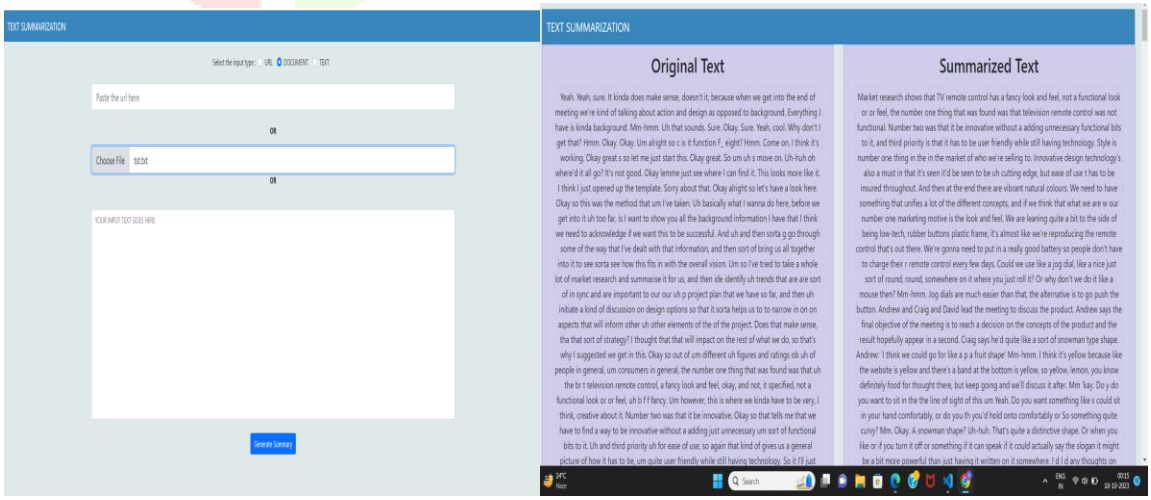
Fig 2. FRONT-END

We have tested this project for different inputs like urls, documents like pdf's, docx , .txt files etc and the text. And the beloe are the different outputs:

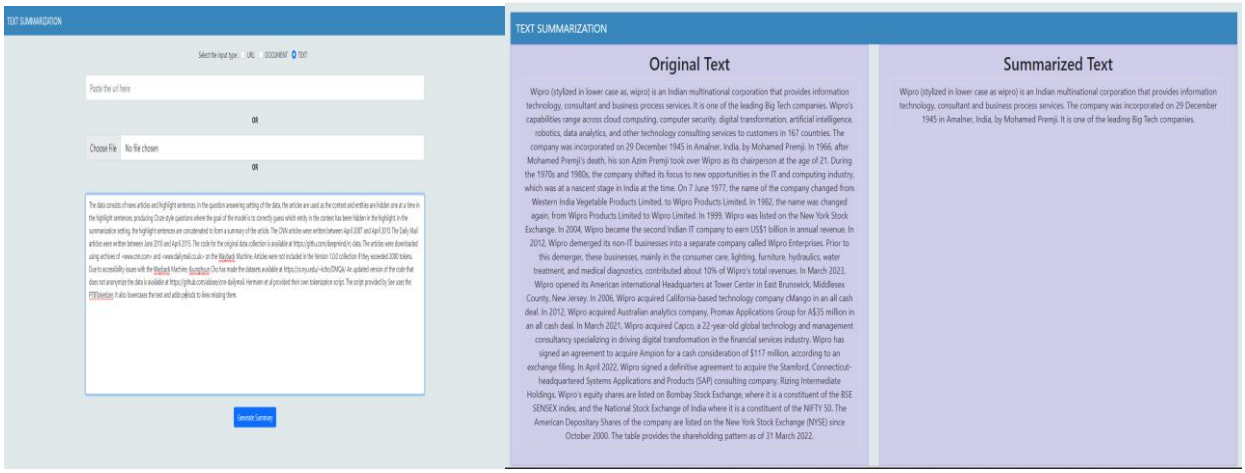
URL AS INPUT:



DOCUMENT AS INPUT(TXT FILE):



TEXT AS INPUT:



We have implemented experiments on the CNN-DailyMail dataset. The CNN / DailyMail Dataset is an English-language dataset containing just over 300k unique news articles as written by journalists at CNN and the Daily Mail. The current version supports both extractive and abstractive summarization, though the original version was created for machine reading and comprehension and abstractive question answering.

Data Fields

- `id`: a string containing the heximal formated SHA1 hash of the url where the story was retrieved from
- `article`: a string containing the body of the news article
- `highlights`: a string containing the highlight of the article as written by the article author

Data Splits

The CNN/DailyMail dataset has 3 splits: *train*, *validation*, and *test*. Below are the statistics for Version 3.0.0 of the dataset.

Dataset Split	Number of Instances in Split
Train	287,113
Validation	13,368
Test	11,490

Fig 3: Dataset Descripiton

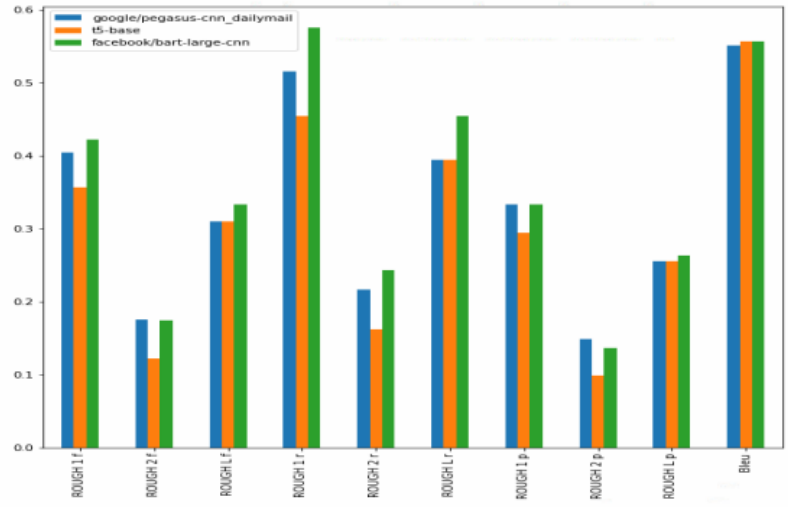


Fig. 4: Performance of three pre-trained models using CNN-dailymail dataset.

	ROUGE									BLEU
	ROUGE-1			ROUGE-2			ROUGE-L			
	R	P	F1	R	P	F1	R	P	F1	
google/pegasus-cnn-dailymail	40.22	32.15	34.60	17.64	13.98	14.97	37.51	30.06	32.32	55.09
T5-base	33.96	30.35	31.10	13.37	11.67	12.00	31.43	28.10	28.79	55.70
facebook/bart-large-cnn	**41.76**	**33.69**	**36.45**	**18.67**	**14.60**	**15.92**	**38.84**	**31.36**	**33.92**	**55.71**

Table 1: Observations of ROUGE and BLEU performance using CNN-dailymail dataset.

	ROUGE									BLEU
	ROUGE-1			ROUGE-2			ROUGE-L			
	R	P	F1	R	P	F1	R	P	F1	
google/pegasus-cnn-dailymail	38.26	25.12	28.39	10.69	7.24	8.05	33.76	22.63	25.36	24.59
T5-base	33.11	26.44	27.88	9.16	7.66	7.72	29.38	23.60	24.82	**26.54**
facebook/bart-large-cnn	**40.63**	**30.57**	**33.02**	**13.16**	**10.10**	**10.58**	**36.38**	**27.69**	**29.78**	25.95

Table 2: Observations of ROUGE and BLEU performance using SAMSum dataset.

	ROUGE									BLEU
	ROUGE-1			ROUGE-2			ROUGE-L			
	R	P	F1	R	P	F1	R	P	F1	
google/pegasus-cnn-dailymail	**26.30**	**48.41**	**30.95**	**11.59**	24.82	**13.99**	**22.95**	42.78	**27.25**	**53.81**
T5-base	21.71	39.45	26.29	7.31	17.71	9.73	18.77	34.56	22.91	54.11
facebook/bart-large-cnn	21.43	46.65	27.82	9.20	**24.99**	12.52	19.53	**42.84**	25.47	58.41

Table 3: Observations of ROUGE and BLEU performance using BillSum dataset.

VI. CONCLUSION

In conclusion, this project successfully developed an end-to-end multi-document abstractive text summarization system with a user-friendly web interface. The system uniquely overcomes input format limitations by enabling summarization of content sourced from diverse documents like webpages, PDFs, text files and DOCX documents. Advanced deep learning models and transformers power the core summarization engine to produce high quality summaries going beyond extraction to generate new phrases and sentences capturing semantic essence. The web portal allows easy interaction from uploading files to obtaining the condensed summary output along with the original text for quick review.

Additional features like tracking past summaries, copying output text and saving summaries in multiple file types further improve user experience. The modular architecture ensures different components like UI, input handlers, summarizer and output can be isolated, modified or extended without impacting other modules. Python enables rapid prototyping with state-of-the-art libraries integrated for web development, CPU/GPU-based deep learning and NLP tasks. The project delivered a usable product combining the strengths of various technologies

while exploring modern solutions to text summarization based on neural networks and transformers.

Future work involves more detailed user studies for qualitative feedback, evaluating summarization quality through metrics like ROUGE and Jensen-Shannon Divergence, exploring extractive summarization methods, and enhancements like multi-language support. On the whole, the project provided great exposure to architecting and implementing an end-to-end NLP pipeline fused with web development.

VII. REFERENCES:

• BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov,Luke Zettlemoyer

• BART Model for Text Summarization : An Analytical Survey and Review Mr. Chandrashekhar Mankar, Adarsh Mundada2 , Sahil Nagrale3 , Pavan Malviya4 , Aniket Sangle5 , Manoj Navrange6

• Y. Chen and Q. Song, "News Text Summarization Method based on BART-TextRank Model," 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC).

• A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization AYESHA AYUB SYED1 , FORD LUMBAN GAOL 1 , (Senior Member, IEEE), AND TOKURO MATSUO2,3, (Member, IEEE)

• Review of automatic text summarization techniques & methods Adhika Pramita Widya sari, , Supriadi Rustad a , Guruh Fajar Shidik a , Edi Noersasongko a , Abdul Syukur a , Affandi Affandi

• A Survey of Text Summarization Extractive Techniques,JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE, VOL. 2, NO. 3, AUGUST 2020

• Automatic Text Summarization Methods: A Comprehensive Review Divakar Yadav1 ,Jalpa Desai, Arun Kumar Yadav2

• A Survey on Automatic Text Summarization Techniques S Hima Bindu Sri and Sushma Rani Dutta 2021 J. Phys.: Conf. Ser. 2040 012044