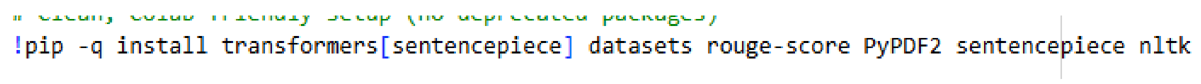
TextMorph: Advanced Text Summarization and Paraphrasing

Datasets used :

1. Andrew S. Tanenbaum - Computer Networks.pdf

2. Neuroscience.pdf

Cell-1 :Environment Setup and Configuration



```
# clean, Colab-friendly setup (no deprecated packages)
!pip -q install transformers[sentencepiece] datasets rouge-score PyPDF2 sentencepiece nltk
```

Installs all the essential Python libraries required for text summarization and paraphrasing tasks:

- **transformers[sentencepiece]** → for using advanced pre-trained NLP models like BART, T5, and Pegasus.
- **datasets** → for handling and loading text data efficiently.
- **rouge-score** → for evaluating the quality of summaries.
- **PyPDF2** → for reading and extracting text from PDF files.
- **sentencepiece** → for tokenization used by Transformer models.
- **nltk** → for additional natural language processing tasks such as text cleaning or sentence tokenization.
- Suppressing Warnings and Progress Bars

Cell-2 : Importing Libraries and setting up fundamental helper functions .

This cell imports all the essential libraries and sets up a helper function to handle PDF text extraction — forming the foundation for later text processing and summarization steps.

1. Standard Python Libraries

- **time, math, json** → Used for performance tracking, mathematical calculations, and data handling (saving or loading data).
- **pathlib.Path** → Simplifies file path operations in a clean, object-oriented way.

- **typing (List, Tuple)** → Helps in defining type hints for better code readability and structure.
- **matplotlib.pyplot** → Used for visualizing data or evaluation metrics later (e.g., plotting ROUGE scores).

2. PDF and NLP Libraries

- **PyPDF2** → Enables reading and extracting text content from PDF files.
- **nltk (Natural Language Toolkit)** → Provides tools for basic NLP tasks.
  - nltk.download('punkt') → Downloads a sentence tokenizer model.
  - sent_tokenize → Splits long text into individual sentences for easier summarization and paraphrasing.

3. Hugging Face Transformers

- **pipeline** → Simplifies using pre-trained models (like T5, BART, Pegasus) for summarization and paraphrasing.
- **AutoTokenizer** and **AutoModelForSeq2SeqLM** → Automatically load the correct tokenizer and model architecture for a given pre-trained model.

4. Evaluation Library

- **rouge_score.rouge_scorer** → Used to calculate ROUGE metrics that measure how well the generated summaries match the original text.

5. Helper Function — extract_text_from_pdf()

This custom function reads a specified range of pages from a PDF and extracts their text content.

Cell 3: Defining PDF File Paths and Page Ranges

The PDF files that will be used for text extraction and summarisation are listed in this cell, along with the corresponding page ranges. Additionally, it quickly validates that the files are uploaded correctly and available in the Colab environment.

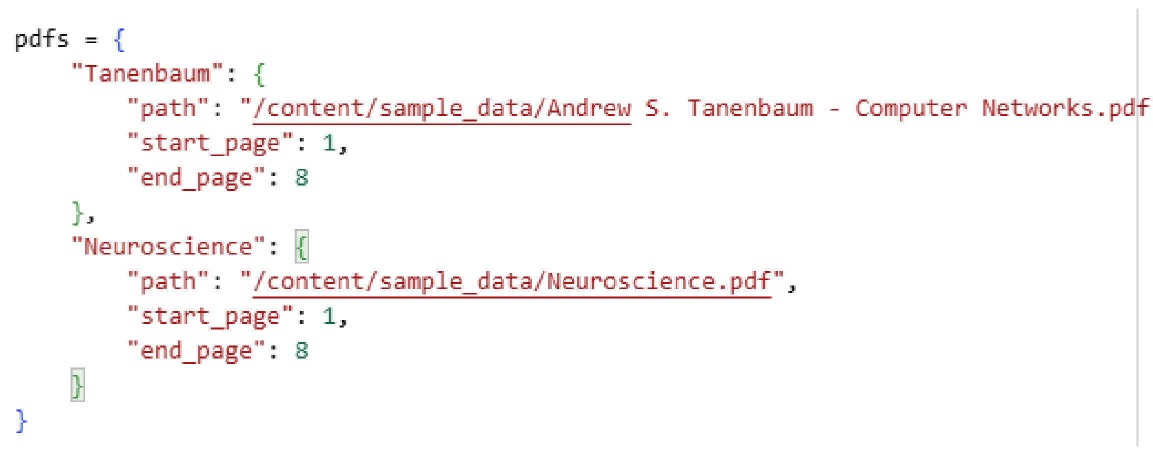
1. PDF File Dictionary Setup

A dictionary named **pdfs** is created to store metadata for each PDF file.

path → The location of the PDF file in the Colab environment.

start_page → The first page number from which text extraction begins (1-indexed).

end_page → The last page number to be included in extraction.



```
pdfs = {
    "Tanenbaum": {
        "path": "/content/sample_data/Andrew S. Tanenbaum - Computer Networks.pdf",
        "start_page": 1,
        "end_page": 8
    },
    "Neuroscience": {
        "path": "/content/sample_data/Neuroscience.pdf",
        "start_page": 1,
        "end_page": 8
    }
}
```

This step ensures all necessary files are properly uploaded before proceeding to text extraction or model-based summarization. It prevents runtime errors by verifying file availability early in the workflow.

Cell 4: Loading and Cleaning Text from PDFs

This cell extracts the text from the PDF files using the previously defined start and end pages, performs basic cleaning, and stores the results in memory for further processing.

This cell ensures that PDF content is:

- Extracted accurately from the desired pages.
- Cleaned of unnecessary whitespace and empty lines.
- Safely stored in variables for summarization or paraphrasing tasks.

Cell 5: Text Chunking Utility

This cell defines a function to break long text into smaller, manageable chunks. Chunking is essential when working with transformer-based models, which often have limits on input length. It also gives an example of how many chunks each document produces.

Purpose

- Ensures **model-friendly input sizes**, preventing errors due to sequence length limits.
- Maintains sentence boundaries to preserve readability and context in summarization tasks.
- Allows processing large documents in smaller, sequential steps.

Cell 6: Model Configuration for Summarization and Paraphrasing

This cell sets up the pre-trained models that will be used for summarization and paraphrasing tasks and checks for GPU availability to optimize processing.

Defines three pre-trained models for **text summarization**:

- **T5** → Flexible transformer for various NLP tasks including summarization.
- **BART** → Encoder-decoder model specialized for sequence-to-sequence tasks.
- **Pegasus-XSum** → Optimized for abstractive summarization on news-style datasets.

Purpose

- Ensures all required models are clearly defined and ready for use.
- Confirms GPU/CPU usage to optimize performance.
- Provides an organized reference to the models for later pipeline creation.

Cell 7: Loading Summarization Models into Pipelines

This cell loads the pre-trained summarization models (T5, BART, Pegasus) and wraps them in Hugging Face **pipelines** for easy text summarization.

Purpose

- Provides a ready-to-use interface for summarizing text with multiple models.
- Ensures models are loaded on the correct device.
- Prepares the environment for efficient, large-scale text summarization.

Cell 8: Loading Paraphrasing Models into Pipelines

This cell loads the pre-trained paraphrasing models (T5, BART, Pegasus) and wraps them in Hugging Face pipelines for easy text paraphrasing.

Purpose

- Provides a ready-to-use interface for generating paraphrased versions of text.
- Ensures models are loaded on the appropriate device.
- Prepares the environment for efficient and consistent paraphrasing operations across different model architectures.

Cell 9: Running Summarization for Each Document and Model

This cell performs the core task of generating summaries for each document using all loaded summarization models. It also tracks performance and prepares results for later evaluation.

Step-by-step process:

1. Chunk the document → Break the text into manageable pieces (10 sentences per chunk).
2. Iterate over models → Summarize the document using each pipeline.
3. Iterate over chunks → Feed each chunk to the model, handling exceptions with alternative parameters (max_length / min_length) if needed.
4. Concatenate summaries → Join chunk-level summaries into a single document summary.
5. Measure time → Record how long the summarization took.
6. Store results → Save summary text, elapsed time, and number of chunks processed.

Purpose

- Produces model-specific summaries for each document.
- Tracks performance metrics like processing time.
- Prepares results for evaluation (e.g., ROUGE scores) or later paraphrasing.
- Ensures long documents are processed safely and efficiently in chunks.

Cell 10: Running Paraphrasing on Generated Summaries

This cell takes the summaries generated in the previous step and produces paraphrased versions using all loaded paraphrasing models. It also tracks processing time and handles long texts safely**.**

Purpose

- Generates alternative expressions of the previously summarized text.
- Handles long summaries safely by chunking.
- Tracks performance metrics for each paraphrasing model.
- Prepares results for comparison, evaluation, or further downstream use.

Cell 11: Evaluating Summarization Performance Using ROUGE

This cell evaluates how well each summarization model performed by calculating ROUGE scores, recording processing time, and measuring summary length. It helps compare the effectiveness and efficiency of different models.

From the output:

- Bart-base often produces longer summaries with higher ROUGE scores, indicating better content coverage.
- Pegasus-xsum tends to generate shorter summaries, sometimes scoring lower on ROUGE metrics.
- t5-base shows moderate performance, balancing length and coverage

Purpose

- Quantitatively evaluates summary quality.
- Compares efficiency (time) and conciseness (length).
- Helps in selecting the most suitable summarization model for the task.

Cell 12: Visualizing Summarization Model Performance

This cell creates visual comparisons of summarization models across ROUGE-1 scores, generation time, and summary length for each document, making it easier to interpret performance at a glance.

Creates a figure with 3 subplots for each document:

- ROUGE-1 F1 score → Measures content quality.
- Generation Time → Shows efficiency of each model.
- Summary Length → Shows conciseness of output.

Bars are labeled with model names and rotated for readability.

plt.tight_layout() ensures plots don't overlap.

Purpose

- Provides an easy-to-read visual summary of model performance.
- Helps quickly identify which models:
    - Produce higher-quality summaries (ROUGE).
    - Are faster or slower (time).
    - Generate longer or shorter summaries (length).
- Useful for making informed decisions about which summarization model to use for a particular document or application.

Cell 13: Display and Save All Summaries and Paraphrases

This cell saves all generated **summaries** and **paraphrases** to text files and creates an interactive **HTML preview** in the notebook for quick inspection.

Purpose

- Persists all generated outputs for later use or sharing.
- Provides an interactive, visually organized preview of results.
- Ensures outputs are easily identifiable by document and model, making review and comparison convenient.

Conclusion

The TextMorph pipeline efficiently extracts text from PDFs, generates summaries using multiple models (T5, BART, Pegasus), and produces paraphrased versions for flexibility. BART provides the most content-rich summaries, T5 balances quality and speed, and Pegasus creates concise outputs. ROUGE evaluation and visualizations show differences in quality, runtime, and length, helping select the best model for each use case. All results are saved and previewed interactively, making this pipeline a complete, end-to-end solution for summarization and paraphrasing of large documents.