Task 2: Text Summarization – Report

Objective

The goal of this task was to develop a summarization system that can automatically condense lengthy news articles into concise and coherent summaries. Two approaches were implemented:

1. **Extractive Summarization** – selects key sentences from the original text.
2. **Abstractive Summarization** – generates new summary sentences using a pre-trained language model.

Dataset Description

I used the **CNN/DailyMail Dataset (version 3.0.0)**, which contains over 287,000 news articles along with their human-written highlights (used as reference summaries). To manage runtime and resources, I used a subset of **100 articles** from the training set for testing and evaluation.

The dataset was accessed via Hugging Face Datasets library:

```
from datasets import load_dataset
dataset = load_dataset("cnn_dailymail", "3.0.0", split="train[:100]")
```

Each record contains:

- article: the full news article text
- highlights: the reference summary

Implementation

1. Preprocessing

Text was preprocessed using the spaCy NLP library:

- Tokenization
- Sentence segmentation

- Stopword removal
- Word frequency calculation for extractive summarization

2. Extractive Summarization (with spaCy)

I implemented a basic frequency-based scoring system to identify the most informative sentences:

- Calculated word frequencies (excluding stopwords)
- Scored sentences based on the sum of frequencies
- Selected the top 3 sentences as the summary

3. Abstractive Summarization (with Hugging Face Transformers)

I used the pre-trained **facebook/bart-large-cnn** model via the Hugging Face pipeline:

```
from transformers import pipeline
summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
```

The model was able to generate fluent, human-like summaries of the input articles.

Results & Evaluation

I ran both extractive and abstractive methods on sample articles. Here's a comparison of one result:

Type	Example Output (Truncated)
Original Article	"...Daniel Radcliffe gains access to a £20M fortune as he turns 18..."
Extractive Summary	"Harry Potter star Daniel Radcliffe gains access to a reported £20M..."
Abstractive Summary	"Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday..."
Reference Summary	"Daniel Radcliffe gets £20M; says he won't spend it on parties or cars."

The extractive summary preserves original sentences, while the abstractive summary is more fluent and closer to the reference summary.

Challenges Faced

- **Dataset access issues**: Hugging Face throttles large loads, so we worked with a subset for practical development.
- **Model loading time**: Abstractive models like BART are large and take time to load.
- **Fine-tuning skipped**: Due to time and hardware constraints, we did not fine-tune models. However, the pre-trained model performed well for this prototype.

Conclusion

I successfully implemented both extractive and abstractive summarization systems using the CNN/DailyMail dataset. While extractive summarization is simpler and faster, abstractive summarization provides more natural and concise results. The system can be further improved by:

- Fine-tuning models on a custom dataset
- Adding ROUGE metric evaluation
- Testing on user-input or live articles

Tools Used

- Google Colab
- Python 3.11
- datasets, transformers, spaCy