TO CREATE A TOOL THAT SUMMARIZES LENGTHY ARTICLES USING NATURAL LANGUAGE PROCESSING TECHNIQUES. INS T R U C T I O NS : TEXT SUMMARIZATION TOOL DELIVERABLE: A PYTHON SCRIPT SHOWCASING INPUT TEXT AND CONCISE SUMMARIES.

To create a tool that summarizes lengthy articles using Natural Language Processing (NLP) techniques, we can use Python along with popular NLP libraries like `nltk`, `spacy`, and `transformers`. The tool will take an input text (article) and generate a concise summary using various NLP techniques like tokenization, sentence segmentation, and pre-trained transformer models.

Here's a step-by-step breakdown of how to create a Text Summarization Tool:

1. Install the Necessary Libraries

We will use libraries like `nltk`, `spacy`, and `transformers` to facilitate text processing and summarization.

bash

Copy code

```bash
pip install nltk spacy transformers python -m spacy download en_core_web_sm
```

2. Import the Libraries

We will import the necessary Python libraries.

python

Copy code

```python
import nltk import spacy from transformers import pipeline from nltk.tokenize import sent_tokenize
```

3. Download Necessary Resources

For sentence tokenization, we need to download resources from NLTK.

python

Copy code

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

4. Text Summarization Function

Now, let's define the core function to summarize text. We'll start by using an extractive summarization approach where we select the most important sentences from the article, then move on to a transformer-based model for abstractive summarization.

Extractive Summarization:

We will extract the most important sentences based on sentence frequency and importance. For simplicity, we can use the sentence tokenizer from `nltk` to break the text into sentences.

python

Copy code

```
def extractive_summary(text): sentences = sent_tokenize(text) # Frequency distribution of words in the text
word_frequencies = {} for word in nltk.word_tokenize(text.lower()): if word.isalnum():
word_frequencies[word] = word_frequencies.get(word, 0) + 1 # Find the most important sentences based on
word frequency sentence_scores = {} for sentence in sentences: for word in
nltk.word_tokenize(sentence.lower()): if word in word_frequencies.keys(): sentence_scores[sentence] =
sentence_scores.get(sentence, 0) + word_frequencies[word] # Sort sentences based on scores and select the top
5 summarized_sentences = sorted(sentence_scores, key=sentence_scores.get, reverse=True)[:5] return '
'.join(summarized_sentences)
```

Abstractive Summarization:

For a more advanced summarization, we use a pre-trained model from Hugging Face's `transformers` library. The pipeline we will use is the `summarization` pipeline, which can generate concise summaries of lengthy articles.

python

Copy code

```
def abstractive_summary(text): summarizer = pipeline("summarization") return summarizer(text,
max_length=150, min_length=50, do_sample=False)[0]['summary_text']
```

5. Main Function

We'll combine both extractive and abstractive summarization methods. The user can choose between them or use both in sequence.

python

Copy code

```
def summarize_text(text, method="both"): if method == "extractive": return extractive_summary(text) elif
method == "abstractive": return abstractive_summary(text) elif method == "both": return { "Extractive
Summary": extractive_summary(text), "Abstractive Summary": abstractive_summary(text) }
```

6. Testing the Tool

Finally, we'll test our tool with a sample article to check both summarization methods.

python

Copy code

```
if __name__ == "__main__": article = """ Natural Language Processing (NLP) is a field of Artificial Intelligence that focuses on enabling computers to understand, interpret, and generate human language. It involves various challenges such as language modeling, sentiment analysis, machine translation, and speech recognition. Over the years, NLP has achieved significant milestones, with the development of large language models such as OpenAI's GPT-3 and Google's BERT. The application of NLP is vast and spans several industries. In healthcare, NLP can assist in analyzing medical records and predicting patient outcomes. In finance, it can automate tasks such as customer service inquiries and market sentiment analysis. Additionally, NLP powers technologies like chatbots, virtual assistants, and voice recognition systems, which are now commonplace in our daily lives. Despite its advancements, NLP still faces challenges, particularly when it comes to understanding context, sarcasm, and the nuances of human language. Researchers are actively working on improving models to make them more efficient and capable of handling complex tasks. As NLP continues to evolve, it holds the potential to revolutionize how humans interact with machines, making it a field of great interest for both academia and industry. """ # Get summary based on user choice summary = summarize_text(article, method="both") print("Extractive Summary:") print(summary["Extractive Summary"]) print("\nAbstractive Summary:") print(summary["Abstractive Summary"])
```

7. Explanation of Code

- **Extractive Summarization:**
    - We tokenize the article into sentences and words.
    - We calculate the frequency of each word and use this to score sentences.
    - The most important sentences (those with the highest word frequency) are selected as the summary.
- **Abstractive Summarization:**
    - This method uses Hugging Face's pre-trained models (`transformers` library) to generate a summary that may differ more significantly from the original text. It works by rephrasing the content and making it more concise.

8. Conclusion

This script is an example of a simple yet powerful text summarization tool that uses both extractive and abstractive methods. It can handle lengthy articles and produce concise summaries, making it a valuable tool for quickly digesting large amounts of information. You can further enhance this tool by implementing additional techniques, such as abstractive summarization with fine-tuning or domain-specific models.