# Text Summarizer

Group members:

Eman Nadeem | 2022-BS-SE-066

Laiba Zafar | 2022-BS-SE-126

Maham Siddique | 2022-BS-SE-129

Dur e Shahwar | 2022-BS-SE-078

## Introduction

A **text summarizer** is a tool or software that automatically condenses a long piece of text into a shorter version, retaining the key information, main ideas, or essential points. It helps readers quickly grasp the core content of the original material without needing to go through the entire text.

## Types of Text Summarizer

Extractive Summarization:

Selects and extracts important sentences from the original text and join them together to form a summary. It doesn't generate new sentences but pulls from the original content.

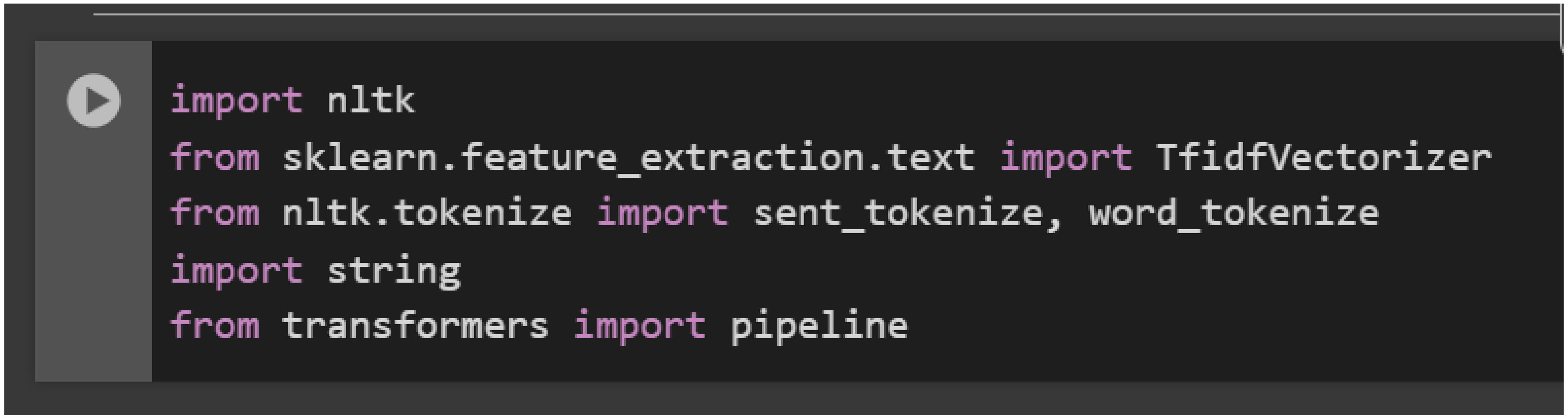
Abstractive Summarization:

Generates new sentences, paraphrases the original text. It tries to understand the underlying meaning and rephrases it, making the summary more concise and readable, while still retaining the key ideas.

## LIBRARIES

- import nltk
- from sklearn.feature_extraction.text import TfidfVectorizer
- from nltk.tokenize import sent_tokenize, word_tokenize
- import string
- from transformers import pipeline

# Libraries



```python
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize, word_tokenize
import string
from transformers import pipeline
```

## TEXT PREPROCESSING

### Tokenization:

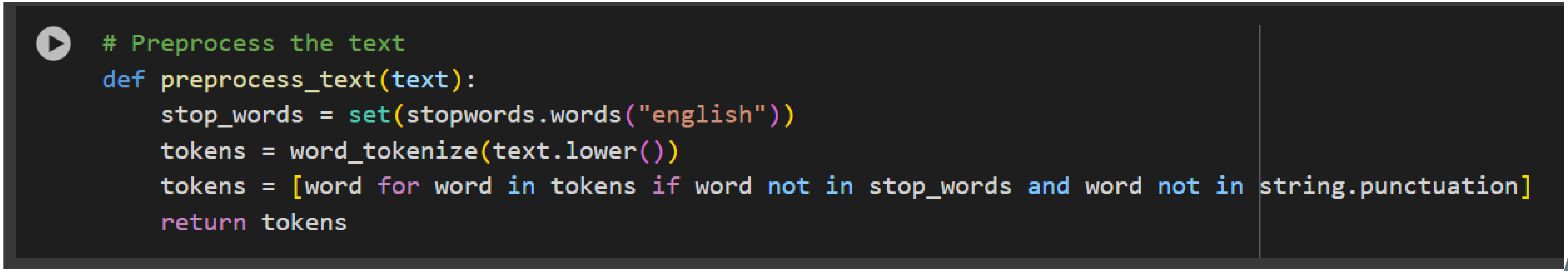
Tokenization is the process of splitting a piece of text into smaller units, called tokens.

### Stopwords:

Stopwords are common words that are often filtered out during text processing because they don't carry significant meaning or value in text analysis

### Lowercasing:

Process of converting all the characters in a text to lowercase letters.



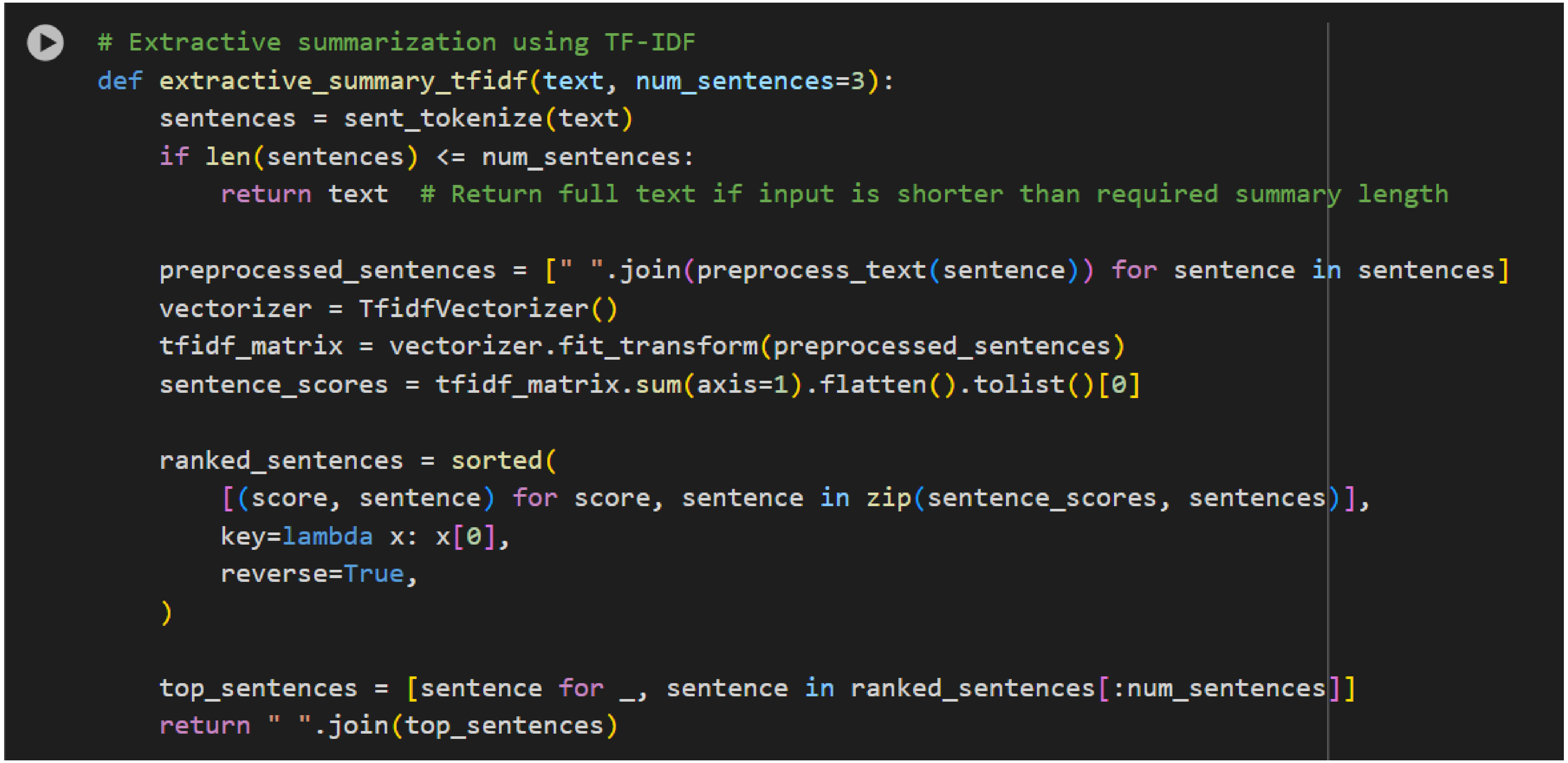
```python
# Preprocess the text
def preprocess_text(text):
    stop_words = set(stopwords.words("english"))
    tokens = word_tokenize(text.lower())
    tokens = [word for word in tokens if word not in stop_words and word not in string.punctuation]
    return tokens
```

## Extractive Summarization

Function for extractive summarization using TF-IDF:

TF-IDF Calculation: Using `TfidfVectorizer` to compute the term frequencies and inverse document frequencies for the sentences in the text.

Ranking Sentences: You compute a score for each sentence based on the sum of its TF-IDF values and then sort the sentences by their scores to select the top sentences for the summary.



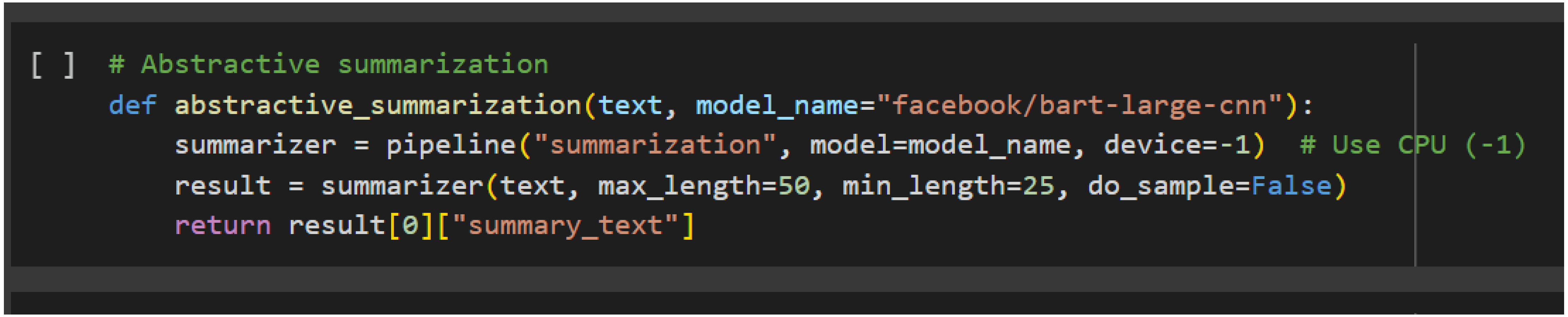
```python
# Extractive summarization using TF-IDF
def extractive_summary_tfidf(text, num_sentences=3):
    sentences = sent_tokenize(text)
    if len(sentences) <= num_sentences:
        return text  # Return full text if input is shorter than required summary length

    preprocessed_sentences = [" ".join(preprocess_text(sentence)) for sentence in sentences]
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(preprocessed_sentences)
    sentence_scores = tfidf_matrix.sum(axis=1).flatten().tolist()[0]

    ranked_sentences = sorted(
        [(score, sentence) for score, sentence in zip(sentence_scores, sentences)],
        key=lambda x: x[0],
        reverse=True,
    )

    top_sentences = [sentence for _, sentence in ranked_sentences[:num_sentences]]
    return " ".join(top_sentences)
```

## Abstractive Summarization

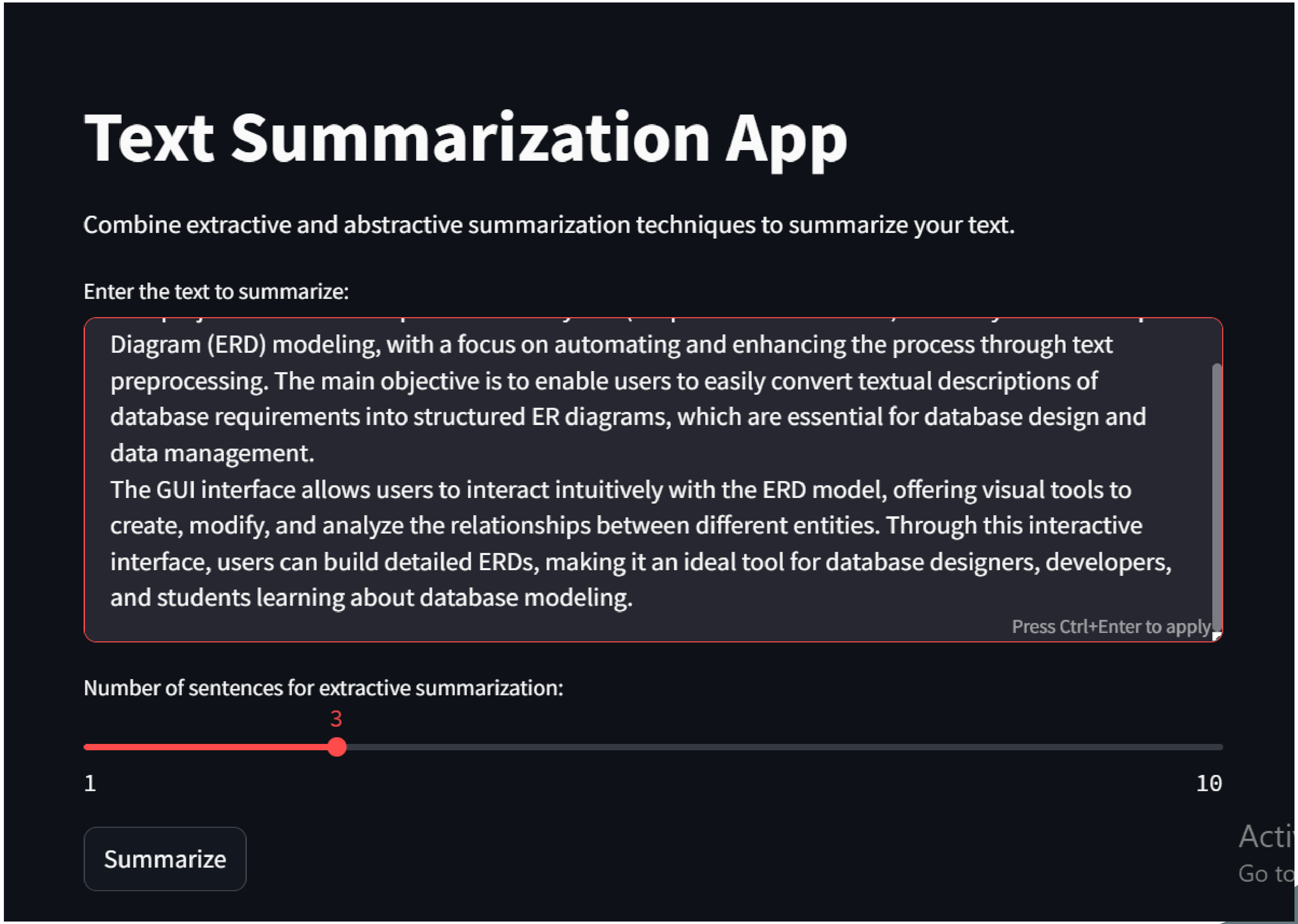
Abstractive summarization is using the Hugging Face **`transformers`** library.

BART (Bidirectional and Auto-Regressive Transformers)

Model Selection: The model **`"facebook/bart-large-cnn"`** is a pre-trained model designed specifically for summarization tasks.



```
[ ]  # Abstractive summarization
     def abstractive_summarization(text, model_name="facebook/bart-large-cnn"):
         summarizer = pipeline("summarization", model=model_name, device=-1)  # Use CPU (-1)
         result = summarizer(text, max_length=50, min_length=25, do_sample=False)
         return result[0]["summary_text"]
```



# Text Summarization App

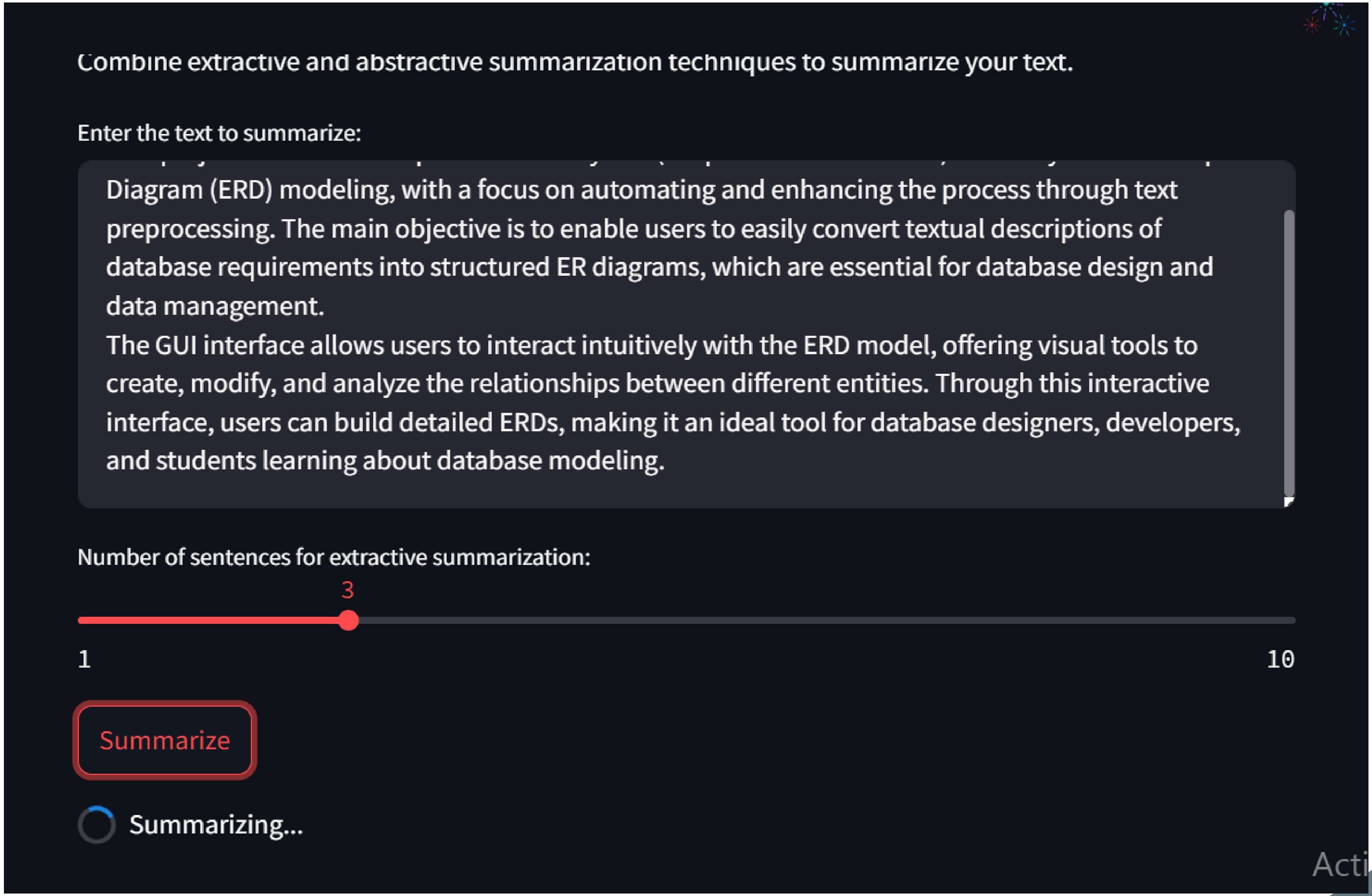
Combine extractive and abstractive summarization techniques to summarize your text.

Enter the text to summarize:

Diagram (ERD) modeling, with a focus on automating and enhancing the process through text preprocessing. The main objective is to enable users to easily convert textual descriptions of database requirements into structured ER diagrams, which are essential for database design and data management.
The GUI interface allows users to interact intuitively with the ERD model, offering visual tools to create, modify, and analyze the relationships between different entities. Through this interactive interface, users can build detailed ERDs, making it an ideal tool for database designers, developers, and students learning about database modeling.

Press Ctrl+Enter to apply

Number of sentences for extractive summarization: 3 (1–10)


Combine extractive and abstractive summarization techniques to summarize your text.

Enter the text to summarize:

Diagram (ERD) modeling, with a focus on automating and enhancing the process through text preprocessing. The main objective is to enable users to easily convert textual descriptions of database requirements into structured ER diagrams, which are essential for database design and data management.
The GUI interface allows users to interact intuitively with the ERD model, offering visual tools to create, modify, and analyze the relationships between different entities. Through this interactive interface, users can build detailed ERDs, making it an ideal tool for database designers, developers, and students learning about database modeling.

Number of sentences for extractive summarization:

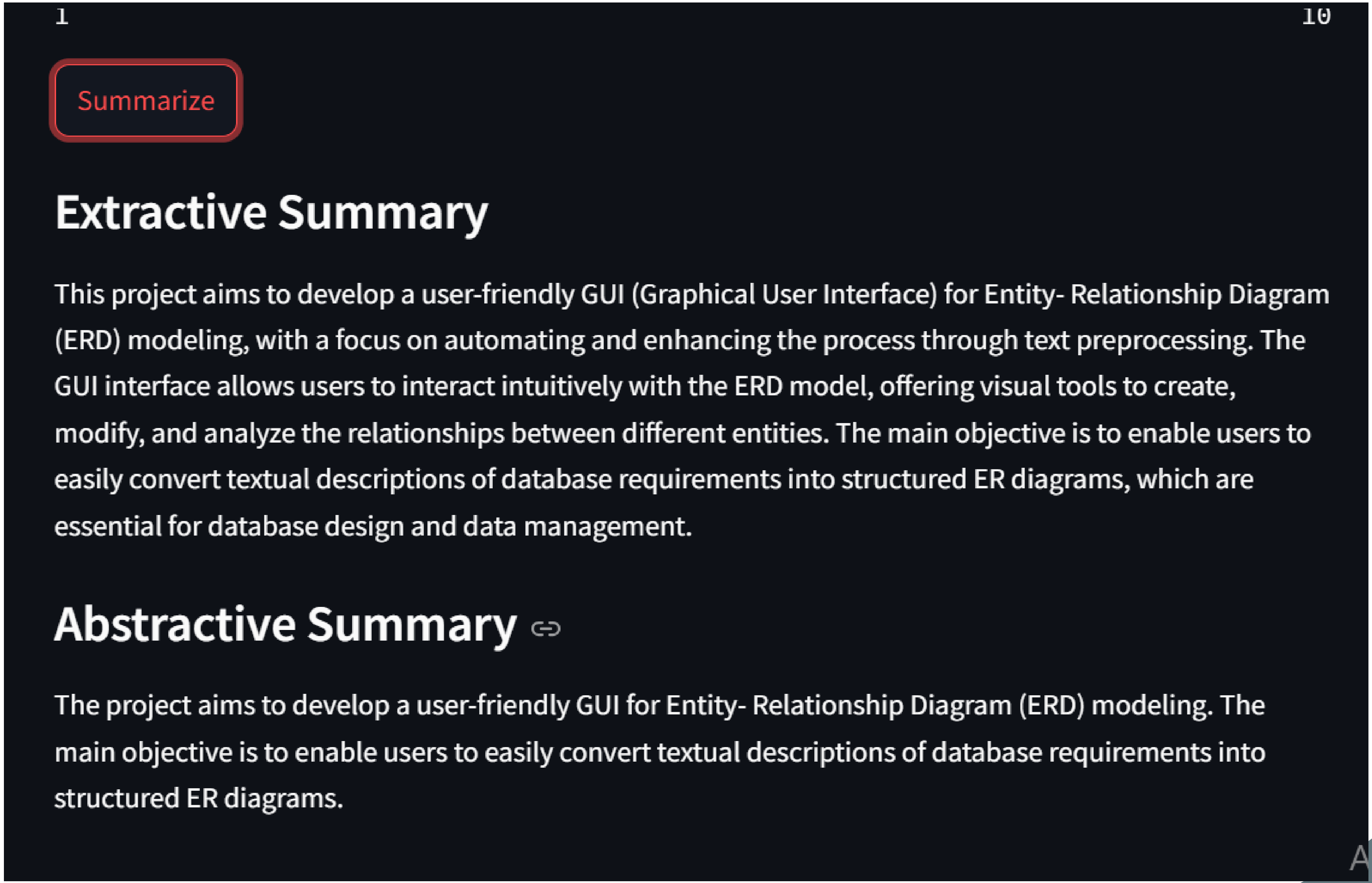
3

1 10

Summarize

Summarizing...

Acti



## Extractive Summary

This project aims to develop a user-friendly GUI (Graphical User Interface) for Entity- Relationship Diagram (ERD) modeling, with a focus on automating and enhancing the process through text preprocessing. The GUI interface allows users to interact intuitively with the ERD model, offering visual tools to create, modify, and analyze the relationships between different entities. The main objective is to enable users to easily convert textual descriptions of database requirements into structured ER diagrams, which are essential for database design and data management.

## Abstractive Summary

The project aims to develop a user-friendly GUI for Entity- Relationship Diagram (ERD) modeling. The main objective is to enable users to easily convert textual descriptions of database requirements into structured ER diagrams.

## Uses of Text Summmarization

- Education: Create concise summaries of textbooks, lectures, and study materials.
- News and Media: Quickly condense news articles for easier reading.
- Healthcare: Summarize medical records, clinical trials, and patient histories.
- Social Media: Track trends and summarize posts or comments.
- Research and Academia: Condense academic papers, literature reviews, and research findings.
- Business: Summarize market reports and competitor analysis.

## Text Summarizer

### Pros

- Time-Saving: Quickly condenses large texts.
- Efficient Information Retrieval: Quickly highlights key information.
- Scalability: Handles large datasets effectively.
- Improves Productivity: Saves time in reading and analyzing texts.

### Cons

- Risk of Misrepresentation: Abstractive models can alter meaning.
- Oversimplification: Important details might be omitted.
- Dependency on Training Data: Quality depends on model data.
- Loss of Context: May miss nuances or critical details.

## Conclusion

Text summarization is a valuable tool for quickly condensing large volumes of text, improving efficiency, and making information more accessible across various industries.