```
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
# This command installs the SpeechRecognition library, which is used for converting speech to text in Python.
#!pip install speechrecognition
```

```
Collecting speechrecognition
  Downloading SpeechRecognition-3.14.1-py3-none-any.whl.metadata (31 kB)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from speechrecognition) (4.12.2)
Downloading SpeechRecognition-3.14.1-py3-none-any.whl (32.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 32.9/32.9 MB 14.8 MB/s eta 0:00:00
Installing collected packages: speechrecognition
Successfully installed speechrecognition-3.14.1
```

```
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
# This command installs the pydub library, which is used for audio processing tasks like converting, slicing, and merging audio files
#!pip install pydub
```

```
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub
Successfully installed pydub-0.25.1
```

```
# This command installs the openai-whisper library, which is a speech recognition model by OpenAI.
# Whisper is capable of transcribing audio into text and supports multiple languages.
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
#!pip install openai-whisper
```

```
Collecting openai-whisper
  Downloading openai-whisper-20240930.tar.gz (800 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 800.5/800.5 kB 12.4 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: numba in /usr/local/lib/python3.11/dist-packages (from openai-whisper) (0.61.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from openai-whisper) (1.26.4)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (from openai-whisper) (2.5.1+cu124)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from openai-whisper) (4.67.1)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.11/dist-packages (from openai-whisper) (10.6.0)
Collecting tiktoken (from openai-whisper)
  Downloading tiktoken-0.9.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.7 kB)
Requirement already satisfied: triton>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from openai-whisper) (3.1.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from triton>=2.0.0->openai-whisper) (3.17.0)
Requirement already satisfied: llvmlite<0.45,>=0.44.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba->openai-whisper
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.11/dist-packages (from tiktoken->openai-whisper) (2024
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.11/dist-packages (from tiktoken->openai-whisper) (2.32
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.11/dist-packages (from torch->openai-whisper)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (3.1.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (2024.10.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch->openai-whisper)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch->openai-whisper)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch->openai-whisper)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch->openai-whisper)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch->openai-whisper)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch->openai-whisper)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch->openai-whisper)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch->openai-whisper
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch->openai-whisper)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch->openai-whisper) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch->openai-w
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.26.0->tiktok
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.26.0->tiktoken->openai-w
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.26.0->tiktoken->op
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.26.0->tiktoken->op
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch->openai-whisper) (3
Downloading tiktoken-0.9.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 47.7 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 1.5 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 95.9 MB/s eta 0:00:00
```

```
'''# Import the AudioSegment class from the pydub library for audio processing tasks.
from pydub import AudioSegment

# Define a function to convert a WAV file to MP3 format.
# Parameters:
# - wav_file: The path to the input WAV file.
# - mp3_file: (Optional) The path to the output MP3 file. If not provided, it will be generated automatically.
def convert_wav_to_mp3(wav_file, mp3_file=None):
    # Load the WAV file as an AudioSegment object.
    audio = AudioSegment.from_wav(wav_file)

    # If the output MP3 file name is not provided, generate it by replacing the .wav extension with .mp3.
    if not mp3_file:
        mp3_file = wav_file.rsplit('.', 1)[0] + ".mp3"

    # Export the audio data as an MP3 file.
    audio.export(mp3_file, format="mp3")

    # Print a message indicating that the conversion is complete.
    print(f"Conversion complete: {mp3_file}")

# Example usage of the function to convert a WAV file to MP3 format.
convert_wav_to_mp3("/content/converted_audio.wav")'''
```

```
'''# Import the whisper library for speech recognition and transcription tasks.
import whisper

# Define a function to transcribe audio files using the Whisper model.
# Parameters:
# - audio_file: The path to the input audio file (e.g., .mp3 or .wav).
# - model_size: The size of the Whisper model to use (default is "base").
#   Other options include "tiny", "small", "medium", and "large" for different accuracy and speed trade-offs.
def transcribe_audio(audio_file, model_size="base"):
    # Load the Whisper model of the specified size.
    model = whisper.load_model(model_size)

    # Transcribe the audio file and store the result.
    result = model.transcribe(audio_file)

    # Generate a file name for saving the transcription by replacing the original file extension with "_transcript.txt".
    transcript_file = audio_file.rsplit('.', 1)[0] + "_transcript.txt"

    # Save the transcription text to a file in UTF-8 encoding.
    with open(transcript_file, "w", encoding="utf-8") as f:
        f.write(result["text"])

    # Print a message indicating where the transcription has been saved.
    print(f"Transcription saved to {transcript_file}")

    # Return the transcription text.
    return result["text"]

# Example usage of the function:
# Specify the path to the audio file to be transcribed.
audio_path = "/content/converted_audio.mp3"  # Change this to your actual file path

# Call the transcription function and print the transcript.
transcript = transcribe_audio(audio_path)
print("Transcript:", transcript)

```

```
'''# Install the pydub library, which is used for audio processing tasks like converting, slicing, and merging audio files.
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
!pip install pydub

# Install the ffmpeg package, which is a powerful tool for handling audio and video files.
# ffmpeg is required by pydub to handle various audio formats (e.g., MP3 to WAV conversion).
# The 'apt install' command is used to install packages on Debian-based Linux systems, like Ubuntu and Google Colab.
!apt install ffmpeg

```

```
Requirement already satisfied: pydub in /usr/local/lib/python3.11/dist-packages (0.25.1)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
```

```
'''# Install the moviepy library, which is used for video editing tasks like cutting, concatenating, and converting videos.
# It also supports extracting and manipulating audio from video files.
```

```
# The exclamation mark (!) is used to run shell commands directly from Jupyter Notebook or Google Colab.
!pip install moviepy

```

```
Requirement already satisfied: moviepy in /usr/local/lib/python3.11/dist-packages (1.0.3)
Requirement already satisfied: decorator<5.0,>=4.0.2 in /usr/local/lib/python3.11/dist-packages (from moviepy) (4.4.2)
Requirement already satisfied: imageio<3.0,>=2.5 in /usr/local/lib/python3.11/dist-packages (from moviepy) (2.37.0)
Requirement already satisfied: imageio_ffmpeg>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from moviepy) (0.6.0)
Requirement already satisfied: tqdm<5.0,>=4.11.2 in /usr/local/lib/python3.11/dist-packages (from moviepy) (4.67.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.11/dist-packages (from moviepy) (1.26.4)
Requirement already satisfied: requests<3.0,>=2.8.1 in /usr/local/lib/python3.11/dist-packages (from moviepy) (2.32.3)
Requirement already satisfied: proglog<=1.0.0 in /usr/local/lib/python3.11/dist-packages (from moviepy) (0.1.10)
Requirement already satisfied: pillow>=8.3.2 in /usr/local/lib/python3.11/dist-packages (from imageio<3.0,>=2.5->moviepy) (11.1.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.8.1->movie
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.8.1->moviepy) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.8.1->moviepy) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.8.1->moviepy) (2
```

```
'''# Import the VideoFileClip class from the moviepy library for video processing tasks.
from moviepy.editor import VideoFileClip

# Define the path to the video file that you want to process.
video_path = "/content/videoplayback.mp4"

# Load the video file into a VideoFileClip object.
video = VideoFileClip(video_path)

# Extract the audio track from the video.
audio = video.audio

# Define the path and format for the extracted audio file.
# Here, we save the audio in MP3 format, but you can change it to WAV or other formats if needed.
audio_path = "extracted_audio.mp3"
audio.write_audiofile(audio_path)

# Print a message indicating that the audio extraction is complete and specify the file name.
print(f"Audio extracted and saved as {audio_path}")

```

```
WARNING:py.warnings:/usr/local/lib/python3.11/dist-packages/moviepy/video/io/sliders.py:61: SyntaxWarning: "is" with a literal. Did
  if event.key is 'enter':

MoviePy - Writing audio in extracted_audio.mp3
MoviePy - Done.
Audio extracted and saved as extracted_audio.mp3
```

```
import IPython.display as ipd

# Play the extracted audio
ipd.Audio("/content/extracted_audio.mp3")
```

```
from pydub import AudioSegment

# Load the audio file
audio = AudioSegment.from_mp3("/content/extracted_audio.mp3")

# Get the duration of the audio in milliseconds
duration = len(audio)

# Calculate the midpoint
midpoint = duration // 2

# Split the audio into two halves
first_half = audio[:midpoint]
second_half = audio[midpoint:]

# Export the two halves as separate files
first_half.export("/content/first_half.mp3", format="mp3")
second_half.export("/content/second_half.mp3", format="mp3")

print("Audio has been split into two halves successfully!")
```

```
Audio has been split into two halves successfully!
```

```
import IPython.display as ipd

# Play the first half
print("Playing the first half:")
ipd.display(ipd.Audio("/content/first_half.mp3"))

# Play the second half
print("Playing the second half:")
ipd.display(ipd.Audio("/content/second_half.mp3"))

```

```
Playing the first half:



Playing the second half:
```

```
import whisper

def transcribe_audio(audio_file, model_size="base"):
    # Load the Whisper model
    model = whisper.load_model(model_size)

    # Transcribe the audio file
    result = model.transcribe(audio_file)

    # Save transcript to a file
    transcript_file = audio_file.rsplit('.', 1)[0] + "_transcript.txt"
    with open(transcript_file, "w", encoding="utf-8") as f:
        f.write(result["text"])

    print(f"Transcription saved to {transcript_file}")
    return result["text"]

# Example usage
audio_path = "/content/first_half.mp3"  # Change this to your actual file
transcript = transcribe_audio(audio_path)
print("Transcript:",transcript)
```

```
WARNING:py.warnings:/usr/local/lib/python3.11/dist-packages/whisper/__init__.py:150: FutureWarning: You are using `torch.load` with
  checkpoint = torch.load(fp, map_location=device)

WARNING:py.warnings:/usr/local/lib/python3.11/dist-packages/whisper/transcribe.py:126: UserWarning: FP16 is not supported on CPU; u
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")

Transcription saved to /content/first_half_transcript.txt
Transcript:  So, sir, we know that India has seen a huge revolution with digital payments. We all thought that India is a place, at
```

```
import whisper

def transcribe_audio(audio_file, model_size="base"):
    # Load the Whisper model
    model = whisper.load_model(model_size)

    # Transcribe the audio file
    result = model.transcribe(audio_file)

    # Save transcript to a file
    transcript_file = audio_file.rsplit('.', 1)[0] + "_transcript.txt"
    with open(transcript_file, "w", encoding="utf-8") as f:
        f.write(result["text"])

    print(f"Transcription saved to {transcript_file}")
    return result["text"]

# Example usage
audio_path = "/content/second_half.mp3"  # Change this to your actual file
transcript = transcribe_audio(audio_path)
print("Transcript:",transcript)
```

```
WARNING:py.warnings:/usr/local/lib/python3.11/dist-packages/whisper/__init__.py:150: FutureWarning: You are using `torch.load` with
  checkpoint = torch.load(fp, map_location=device)

WARNING:py.warnings:/usr/local/lib/python3.11/dist-packages/whisper/transcribe.py:126: UserWarning: FP16 is not supported on CPU; u
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")

Transcription saved to /content/second_half_transcript.txt
Transcript:  Tintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintintinti
```

```
# prompt: Generate code to read the video url and extract the transcript of a video

# Install necessary libraries
!pip install youtube-transcript-api

from youtube_transcript_api import YouTubeTranscriptApi

def get_transcript(video_url):
    """
    Extracts the transcript from a YouTube video URL.

    Args:
        video_url: The URL of the YouTube video.

    Returns:
        A list of dictionaries, where each dictionary represents a segment of the transcript
        with 'text' and 'start' keys, or None if an error occurs.
    """

    try:
        # Extract video ID from URL
        video_id = video_url.split("v=")[-1]
        if "&" in video_id:
          video_id = video_id[:video_id.index("&")]

        # Get transcript
        transcript = YouTubeTranscriptApi.get_transcript(video_id)
        return transcript

    except Exception as e:
        print(f"Error: {e}")
        return None

# Example usage
video_url = input("Enter the YouTube video URL: ")
transcript = get_transcript(video_url)

if transcript:
  print("\nTranscript:")
  for segment in transcript:
    print(f"{segment['text']}")
```

```
#This is the final code
```

```
!pip install youtube_transcript_api
```

```
Collecting youtube_transcript_api
  Downloading youtube_transcript_api-0.6.3-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: defusedxml<0.8.0,>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from youtube_transcript_api) (0
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from youtube_transcript_api) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcrip
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api) (3.16
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api
Downloading youtube_transcript_api-0.6.3-py3-none-any.whl (622 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 622.3/622.3 kB 12.8 MB/s eta 0:00:00
Installing collected packages: youtube_transcript_api
Successfully installed youtube_transcript_api-0.6.3
```

```
pip install pytube speechrecognition pydub
```

```
Collecting pytube
  Downloading pytube-15.0.0-py3-none-any.whl.metadata (5.0 kB)
Collecting speechrecognition
  Downloading SpeechRecognition-3.14.1-py3-none-any.whl.metadata (31 kB)
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from speechrecognition) (4.12.2)
Downloading pytube-15.0.0-py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.6/57.6 kB 3.2 MB/s eta 0:00:00
Downloading SpeechRecognition-3.14.1-py3-none-any.whl (32.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 32.9/32.9 MB 27.2 MB/s eta 0:00:00
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub, speechrecognition, pytube
Successfully installed pydub-0.25.1 pytube-15.0.0 speechrecognition-3.14.1
```

```
import re
import urllib.parse
```

```
import requests
from youtube_transcript_api import YouTubeTranscriptApi
from pytube import YouTube
import speech_recognition as sr
from pydub import AudioSegment
import os

def extract_video_id(video_url):
    """
    Extracts the YouTube video ID from various URL formats.
    """
    parsed_url = urllib.parse.urlparse(video_url)
    query_params = urllib.parse.parse_qs(parsed_url.query)

    if "v" in query_params:
        return query_params["v"][0]

    match = re.search(r"(youtu\.be/|youtube\.com/embed/|youtube\.com/shorts/)([\w-]+)", video_url)
    if match:
        return match.group(2)

    return None

def download_audio(video_url):
    """
    Downloads the audio using yt-dlp with cookies and returns the file path.
    """
    try:
        ydl_opts = {
            'format': 'bestaudio/best',
            'outtmpl': 'audio.%(ext)s',
            'cookiefile': 'cookies (1).txt',  # Use the exported cookies
            'postprocessors': [{
                'key': 'FFmpegExtractAudio',
                'preferredcodec': 'mp3',
                'preferredquality': '192',
            }],
        }
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            info = ydl.extract_info(video_url, download=True)
            return "audio.mp3"
    except Exception as e:
        return f"Error downloading audio: {str(e)}"

def convert_audio_to_wav(audio_file):
    """
    Converts the downloaded MP3 audio to WAV format using pydub.
    """
    wav_file = "audio.wav"
    try:
        AudioSegment.from_mp3(audio_file).export(wav_file, format="wav")
        return wav_file
    except Exception as e:
        return f"Error converting to WAV: {str(e)}"

def transcribe_audio(audio_path, chunk_length=30):
    """
    Splits audio into smaller chunks and transcribes each chunk separately.
    Args:
        audio_path (str): Path to the audio file.
        chunk_length (int): Length of each chunk in seconds (default: 30).
    Returns:
        str: Transcribed text from the audio.
    """
    recognizer = sr.Recognizer()
    audio = AudioSegment.from_wav(audio_path)
    total_duration = len(audio) / 1000  # Convert to seconds
    transcribed_text = []

    print("Transcribing audio in chunks...")

    # Split and transcribe audio in chunks
    for start in range(0, int(total_duration), chunk_length):
        end = min(start + chunk_length, int(total_duration))
        chunk = audio[start * 1000:end * 1000]  # Extract chunk in milliseconds
        chunk.export("chunk.wav", format="wav")  # Save chunk temporarily

        with sr.AudioFile("chunk.wav") as source:
            try:
                audio_data = recognizer.record(source)
                text = recognizer.recognize_google(audio_data)
                transcribed_text.append(text)
```

```
            except sr.UnknownValueError:
                transcribed_text.append("[Unintelligible]")
            except sr.RequestError as e:
                return f"Error with the speech recognition service: {str(e)}"

    os.remove("chunk.wav")  # Clean up temporary chunk file
    return "\n".join(transcribed_text)

def get_transcript_unlisted(video_url):
    """
    Tries to fetch the transcript using youtube_transcript_api first,
    then falls back to downloading and transcribing audio if necessary.
    """
    video_id = extract_video_id(video_url)
    if not video_id:
        return "Invalid YouTube URL."

    # Try to fetch transcript using youtube_transcript_api
    try:
        transcript = YouTubeTranscriptApi.get_transcript(video_id)
        return "\n".join([item['text'] for item in transcript])
    except:
        print("Transcript not available via API, attempting audio transcription...")

    # Download and transcribe audio if no transcript is available
    audio_file = download_audio(video_url)
    if "Error" in audio_file:
        return audio_file

    wav_file = convert_audio_to_wav(audio_file)
    if "Error" in wav_file:
        return wav_file

    transcription = transcribe_audio(wav_file)

    # Cleanup temporary files
    os.remove(audio_file)
    os.remove(wav_file)

    return transcription

# Example usage
if __name__ == "__main__":
    video_url = input("Enter the YouTube video URL: ")
    transcript = get_transcript_unlisted(video_url)
    print("\nTranscript:\n", transcript)
```

```
Enter the YouTube video URL: https://youtu.be/sK8SILOM37I
Transcript not available via API, attempting audio transcription...
[youtube] Extracting URL: https://youtu.be/sK8SILOM37I
[youtube] sK8SILOM37I: Downloading webpage
[youtube] sK8SILOM37I: Downloading tv client config
[youtube] sK8SILOM37I: Downloading player b191cf34
[youtube] sK8SILOM37I: Downloading tv player API JSON
[info] sK8SILOM37I: Downloading 1 format(s): 251
[download] Destination: audio.webm
[download] 100% of   39.37MiB in 00:00:05 at 6.69MiB/s
[ExtractAudio] Destination: audio.mp3
Deleting original file audio.webm (pass -k to keep)
Transcribing audio in chunks...

Transcript:
 so sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the b
I think immediately after you play the next big revolution personally I think is an education and the complete homework for this
do you think ncrf plus any people together will be the next big revolution after up in India absolutely and why I think so is bec
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
July 2020 and we recently celebrated the 4th anniversary of NP 2020 ncrf has been brought to implement the intent of 2020 2020 th
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guidel
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
pull up my kitchen to add to my taste buds in a way that it's convenient for me and one fine day you came and you changed my kitc
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will you be not only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we ar
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyin
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for nc
it allows for creditor of all learnings weather in academics orange killing or an experiential learning and all these three kinds
and people who are already skilled or already in the professional area not there the experiential learning would play a big part
increase of technology which has been created by single Department good question so therefore all of us we have to work in tandem
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills y
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of all
```

```
have you already seen the Fallout of this know you can see the photo you can see how many Engineers are there for Designing our b
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework hc
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept pract
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those ar
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 yea
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so th
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test t
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumir
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that w
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessme
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assessm
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and t
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you fe
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-mindec
```

```
pip install yt-dlp
```

```
Collecting yt-dlp
  Downloading yt_dlp-2025.2.19-py3-none-any.whl.metadata (171 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 171.9/171.9 kB 4.3 MB/s eta 0:00:00
Downloading yt_dlp-2025.2.19-py3-none-any.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 32.6 MB/s eta 0:00:00
Installing collected packages: yt-dlp
Successfully installed yt-dlp-2025.2.19
```

```
import yt_dlp
```

```
from pydub import AudioSegment
import os
```

```
import re
from collections import Counter
import nltk
from nltk.tokenize import sent_tokenize
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer

def get_transcript():
    """Prompt user to input the transcript text."""
    print("Enter the transcript (end input with 'END'):")
    lines = []
    while True:
        line = input()
        if line.strip().upper() == "END":
            break
        lines.append(line)
    return "\n".join(lines)

def extract_topics(transcript, num_topics=5):
    """Extract topics from the transcript using TF-IDF and clustering."""
    sentences = sent_tokenize(transcript)
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(sentences)

    kmeans = KMeans(n_clusters=num_topics, random_state=42, n_init=10)
    kmeans.fit(X)

    topic_sentences = {i: [] for i in range(num_topics)}
    for i, label in enumerate(kmeans.labels_):
        topic_sentences[label].append(sentences[i])

    topics = {f"Topic {i+1}": " ".join(topic_sentences[i][:3]) for i in range(num_topics)}
    return topics

def segment_transcript(transcript):
    """Segment the transcript based on extracted topics."""
    topics = extract_topics(transcript)

    segmented_text = ""
    for heading, example_text in topics.items():
        segmented_text += f"\n### {heading}\n- {example_text}...\n"

    return segmented_text

def main():
    nltk.download('punkt')  # Ensure required tokenizer is available
```

```
    transcript = get_transcript()
    segmented_text = segment_transcript(transcript)

    output_file = input("Enter the desired output file name (or press Enter for default: segmented_transcript.txt): ")
    output_file = output_file if output_file else "segmented_transcript.txt"

    with open(output_file, "w", encoding="utf-8") as file:
        file.write(segmented_text)

    print(f"Segmentation complete! Output saved to {output_file}")

if __name__ == "__main__":
    main()

```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Enter the transcript (end input with 'END'):
so sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the best
END
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-1-47d2f9668108> in <cell line: 0>()
     57
     58 if __name__ == "__main__":
---> 59     main()

                              7 frames
/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    577     sep = "*" * 70
    578     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579     raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

```
import re
import nltk
from nltk.tokenize import sent_tokenize
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer

def read_transcript(file_path):
    """Read the transcript from a file."""
    with open(file_path, "r", encoding="utf-8") as file:
        return file.read()

def extract_topics(transcript, num_topics=5):
    """Extract topics from the transcript using TF-IDF and clustering."""
    sentences = sent_tokenize(transcript)
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(sentences)

    kmeans = KMeans(n_clusters=num_topics, random_state=42, n_init=10)
    kmeans.fit(X)

    topic_sentences = {i: [] for i in range(num_topics)}
    for i, label in enumerate(kmeans.labels_):
        topic_sentences[label].append(sentences[i])

    topics = {f"Topic {i+1}": " ".join(topic_sentences[i][:3]) for i in range(num_topics)}
```

```
    return topics

def segment_transcript(transcript):
    """Segment the transcript based on extracted topics."""
    topics = extract_topics(transcript)

    segmented_text = ""
    for heading, example_text in topics.items():
        segmented_text += f"\n### {heading}\n- {example_text}...\n"

    return segmented_text

def main():
    nltk.download('punkt')  # Ensure required tokenizer is available
    input_file = input("Enter the path of the transcript file: ")
    transcript = read_transcript(input_file)
    segmented_text = segment_transcript(transcript)

    output_file = input("Enter the desired output file name (or press Enter for default: segmented_transcript.txt): ")
    output_file = output_file if output_file else "segmented_transcript.txt"

    with open(output_file, "w", encoding="utf-8") as file:
        file.write(segmented_text)

    print(f"Segmentation complete! Output saved to {output_file}")

if __name__ == "__main__":
    main()

```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Enter the path of the transcript file: /content/Kalsi sir transcript.txt
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-2-ec884cafd024> in <cell line: 0>()
     51
     52 if __name__ == "__main__":
---> 53     main()

                              7 frames
/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    577     sep = "*" * 70
    578     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579     raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

```python
import re
import nltk
from nltk.tokenize import sent_tokenize
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer

def read_transcript(file_path):
    """Read the transcript from a file."""
    with open(file_path, "r", encoding="utf-8") as file:
        return file.read()

def extract_topics(transcript, num_topics=5):
    """Extract topics from the transcript using TF-IDF and clustering."""
    nltk.download('punkt')  # Ensure required tokenizer is available
    sentences = sent_tokenize(transcript)
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(sentences)

    kmeans = KMeans(n_clusters=num_topics, random_state=42, n_init=10)
    kmeans.fit(X)

    topic_sentences = {i: [] for i in range(num_topics)}
    for i, label in enumerate(kmeans.labels_):
        topic_sentences[label].append(sentences[i])

    topics = {f"Topic {i+1}": " ".join(topic_sentences[i][:3]) for i in range(num_topics)}
    return topics

def segment_transcript(transcript):
    """Segment the transcript based on extracted topics."""
    topics = extract_topics(transcript)

    segmented_text = ""
    for heading, example_text in topics.items():
        segmented_text += f"\n### {heading}\n- {example_text}...\n"

    return segmented_text

def main():
    nltk.download('punkt')  # Ensure required tokenizer is available
    input_file = input("Enter the path of the transcript file: ")
    transcript = read_transcript(input_file)
    segmented_text = segment_transcript(transcript)

    output_file = input("Enter the desired output file name (or press Enter for default: segmented_transcript.txt): ")
    output_file = output_file if output_file else "segmented_transcript.txt"

    with open(output_file, "w", encoding="utf-8") as file:
        file.write(segmented_text)

    print(f"Segmentation complete! Output saved to {output_file}")

if __name__ == "__main__":
    main()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Enter the path of the transcript file: /content/Kalsi sir transcript.txt
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-5-4884c9bae683> in <cell line: 0>()
     52
     53 if __name__ == "__main__":
---> 54     main()

                              7 frames
/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    577     sep = "*" * 70
    578     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579     raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import nltk
import os
import shutil

# Remove existing nltk_data directory
nltk_data_path = os.path.expanduser('~/nltk_data')
if os.path.exists(nltk_data_path):
    shutil.rmtree(nltk_data_path)

# Download necessary resources again
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import re
import nltk
from nltk.tokenize import sent_tokenize
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer

def read_transcript(file_path):
    """Read the transcript from a file."""
    with open(file_path, "r", encoding="utf-8") as file:
        return file.read()

def extract_topics(transcript, num_topics=5):
    """Extract topics from the transcript using TF-IDF and clustering."""
    nltk.download('punkt')  # Ensure required tokenizer is available
    sentences = sent_tokenize(transcript)
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(sentences)
```

```python

    kmeans = KMeans(n_clusters=num_topics, random_state=42, n_init=10)
    kmeans.fit(X)

    topic_sentences = {i: [] for i in range(num_topics)}
    for i, label in enumerate(kmeans.labels_):
        topic_sentences[label].append(sentences[i])

    topics = {f"Topic {i+1}": " ".join(topic_sentences[i][:3]) for i in range(num_topics)}
    return topics

def segment_transcript(transcript):
    """Segment the transcript based on extracted topics."""
    topics = extract_topics(transcript)

    segmented_text = ""
    for heading, example_text in topics.items():
        segmented_text += f"\n### {heading}\n- {example_text}...\n"

    return segmented_text

def main():
    nltk.download('punkt')  # Ensure required tokenizer is available
    input_file = input("Enter the path of the transcript file: ")
    transcript = read_transcript(input_file)
    segmented_text = segment_transcript(transcript)

    output_file = input("Enter the desired output file name (or press Enter for default: segmented_transcript.txt): ")
    output_file = output_file if output_file else "segmented_transcript.txt"

    with open(output_file, "w", encoding="utf-8") as file:
        file.write(segmented_text)

    print(f"Segmentation complete! Output saved to {output_file}")

if __name__ == "__main__":
    main()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Enter the path of the transcript file: /content/Kalsi sir transcript.txt
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-7-4884c9bae683> in <cell line: 0>()
     52
     53 if __name__ == "__main__":
---> 54     main()

                              7 frames
/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    577     sep = "*" * 70
    578     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579     raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import nltk
print(nltk.data.path)
```

```
['/root/nltk_data', '/usr/nltk_data', '/usr/share/nltk_data', '/usr/lib/nltk_data', '/usr/share/nltk_data', '/usr/local/share/nltk_c
```

```
nltk.data.path.append('/usr/local/share/nltk_data')
```

```
import shutil
import os
import nltk

nltk_data_path = os.path.expanduser('~/nltk_data')
if os.path.exists(nltk_data_path):
    shutil.rmtree(nltk_data_path)

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import os
print(os.listdir(nltk.data.find("tokenizers")))
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-12-82277fac31aa> in <cell line: 0>()
      1 import os
----> 2 print(os.listdir(nltk.data.find("tokenizers")))

/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    554
    555     # Identify the package (i.e. the .zip file) to download.
--> 556     resource_zipname = resource_name.split("/")[1]
    557     if resource_zipname.endswith(".zip"):
    558         resource_zipname = resource_zipname.rpartition(".")[0]

IndexError: list index out of range
```

```
import nltk
print(nltk.data.path)
```

```
['/root/nltk_data', '/usr/nltk_data', '/usr/share/nltk_data', '/usr/lib/nltk_data', '/usr/share/nltk_data', '/usr/local/share/nltk_c
```

```
import os
nltk_data_path = os.path.expanduser("~/nltk_data/tokenizers")
if os.path.exists(nltk_data_path):
    print(os.listdir(nltk_data_path))
else:
    print("NLTK tokenizer directory not found!")
```

```
NLTK tokenizer directory not found!
```

```
import os

nltk_data_path = os.path.expanduser("~/nltk_data/tokenizers")
if not os.path.exists(nltk_data_path):
    os.makedirs(nltk_data_path)

print(f"Created directory: {nltk_data_path}")
```

```
Created directory: /root/nltk_data/tokenizers
```

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import os

nltk_data_path = os.path.expanduser("~/nltk_data/tokenizers")
if os.path.exists(nltk_data_path):
    print(os.listdir(nltk_data_path))
else:
    print("NLTK tokenizer directory still missing!")
```

```
[]
```

```
import nltk
import os

# Set a specific NLTK data directory
nltk_data_dir = os.path.expanduser("~/nltk_data")
if not os.path.exists(nltk_data_dir):
    os.makedirs(nltk_data_dir)

# Append the path manually in case it's not detected
nltk.data.path.append(nltk_data_dir)

# Download the 'punkt' tokenizer
nltk.download('punkt', download_dir=nltk_data_dir)

# Verify that the tokenizer exists now
tokenizer_path = os.path.join(nltk_data_dir, "tokenizers")
print(f"Checking for tokenizers in: {tokenizer_path}")
print(os.listdir(tokenizer_path) if os.path.exists(tokenizer_path) else "Tokenizer directory not found!")
```

```
Checking for tokenizers in: /root/nltk_data/tokenizers
[]
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
import nltk
from nltk.data import find

try:
    punkt_path = find('tokenizers/punkt')
    print(f"'punkt' found at: {punkt_path}")
except LookupError:
    print("Could not find 'punkt' in nltk data.")
```

```
Could not find 'punkt' in nltk data.
```

```
import shutil
import os

nltk_data_path = os.path.expanduser("~/nltk_data")

if os.path.exists(nltk_data_path):
    shutil.rmtree(nltk_data_path)

print("Deleted existing NLTK data. Ready for fresh installation.")
```

```
Deleted existing NLTK data. Ready for fresh installation.
```

```
import nltk
import os

nltk_data_path = os.path.expanduser("~/nltk_data")
os.makedirs(nltk_data_path, exist_ok=True)

nltk.download('punkt', download_dir=nltk_data_path)

print("Downloaded 'punkt' to:", nltk_data_path)
```

```
Downloaded 'punkt' to: /root/nltk_data
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
import os

tokenizer_path = os.path.expanduser("~/nltk_data/tokenizers")
print("Checking for tokenizers in:", tokenizer_path)
print(os.listdir(tokenizer_path) if os.path.exists(tokenizer_path) else "Tokenizer directory not found!")
```

```
Checking for tokenizers in: /root/nltk_data/tokenizers
Tokenizer directory not found!
```

```python
# prompt: segment the transcript from the path with relevant topics

import nltk
from nltk.tokenize import sent_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import os

def read_transcript(file_path):
    """Read the transcript from a file."""
    with open(file_path, "r", encoding="utf-8") as file:
        return file.read()

def extract_topics(transcript, num_topics=5):
    """Extract topics from the transcript using TF-IDF and clustering."""
    try:
        sentences = sent_tokenize(transcript)
        vectorizer = TfidfVectorizer(stop_words='english')
        X = vectorizer.fit_transform(sentences)

        kmeans = KMeans(n_clusters=num_topics, random_state=42, n_init=10)
        kmeans.fit(X)

        topic_sentences = {i: [] for i in range(num_topics)}
        for i, label in enumerate(kmeans.labels_):
            topic_sentences[label].append(sentences[i])

        topics = {f"Topic {i+1}": " ".join(topic_sentences[i][:3]) for i in range(num_topics)}
        return topics
    except Exception as e:
        print(f"An error occurred during topic extraction: {e}")
        return None

def segment_transcript(transcript):
    """Segment the transcript based on extracted topics."""
    topics = extract_topics(transcript)
    if topics is None:
        return "Topic extraction failed."

    segmented_text = ""
    for heading, example_text in topics.items():
        segmented_text += f"\n### {heading}\n- {example_text}...\n"

    return segmented_text

def main():
    nltk.download('punkt', quiet=True) # Download punkt if not present, suppress output
    input_file = input("Enter the path of the transcript file: ")

    if not os.path.exists(input_file):
        print(f"Error: File '{input_file}' not found.")
        return

    transcript = read_transcript(input_file)
    segmented_text = segment_transcript(transcript)

    output_file = input("Enter the desired output file name (or press Enter for default: segmented_transcript.txt): ")
    output_file = output_file if output_file else "segmented_transcript.txt"

    with open(output_file, "w", encoding="utf-8") as file:
        file.write(segmented_text)

    print(f"Segmentation complete! Output saved to {output_file}")

if __name__ == "__main__":
    main()
```

```
Enter the path of the transcript file: /content/Kalsi sir transcript.txt
An error occurred during topic extraction:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
```

```
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/root/nltk_data'
**********************************************************************

Enter the desired output file name (or press Enter for default: segmented_transcript.txt):
Segmentation complete! Output saved to segmented_transcript.txt
```

```
#Another approach
```

```
pip install transformers nltk
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.48.3)
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.17.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.28.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.24.0->trans
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.2
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025.1.3
```

```
import os
import nltk
from transformers import pipeline
from nltk.tokenize import sent_tokenize
import torch

# Download the Punkt tokenizer for sentence splitting if not already downloaded
nltk.download('punkt')

# Define file paths
input_file = '/content/Kalsi sir transcript.txt'        # Replace with your transcript file name
output_file = 'segmented_transcript.txt'

# Load the summarization model for heading generation
summarizer = pipeline("summarization", model="facebook/bart-large-cnn", device=0 if torch.cuda.is_available() else -1)

# Read the transcript
with open(input_file, 'r', encoding='utf-8') as file:
    transcript = file.read()

# Split transcript into sentences
sentences = sent_tokenize(transcript)

# Segment transcript into chunks of approximately 500 words for manageable heading generation
chunk_size = 300
chunks = []
current_chunk = []

for sentence in sentences:
    current_chunk.append(sentence)
    if sum(len(s.split()) for s in current_chunk) >= chunk_size:
        chunks.append(' '.join(current_chunk))
        current_chunk = []

if current_chunk:
    chunks.append(' '.join(current_chunk))

# Generate headings and organize content
segmented_content = ""
for idx, chunk in enumerate(chunks, 1):
    if not chunk.strip():
        continue  # Skip empty chunks

    # Limit chunk to 1024 tokens if it's too large
```

```
    tokens = tokenizer(chunk)['input_ids']
    if len(tokens) > 1024:
        chunk = tokenizer.decode(tokens[:1024])

    # Generate a heading using summarization
    heading = summarizer(chunk, max_length=10, min_length=3, do_sample=False)[0]['summary_text']
    segmented_content += f"### {heading}\n\n{chunk}\n\n"

# Save segmented content to a new file
with open(output_file, 'w', encoding='utf-8') as file:
    file.write(segmented_content)

print(f"Segmented transcript saved to {output_file}")

```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Device set to use cpu
Token indices sequence length is longer than the specified maximum sequence length for this model (3126 > 512). Running this sequenc
Segmented transcript saved to segmented_transcript.txt
```

```
import nltk
nltk.download('all')
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_eng to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_eng.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_rus to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_rus.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package bcp47 to /root/nltk_data...
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_esp.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
```

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("t5-small")
```

```
tokenizer_config.json: 100%  2.32k/2.32k [00:00<00:00, 191kB/s]
spiece.model: 100%  792k/792k [00:00<00:00, 15.5MB/s]
tokenizer.json: 100%  1.39M/1.39M [00:00<00:00, 16.3MB/s]
```

```python
import os
import re
import nltk
from transformers import pipeline, AutoTokenizer
from nltk.tokenize import sent_tokenize
import torch

# Download the Punkt tokenizer for sentence splitting if not already downloaded
nltk.download('punkt')

# Define file paths
input_file = '/content/Kalsi sir transcript.txt'  # Replace with your transcript file name
output_file = 'segmented_transcript.txt'

# Load the summarization model and tokenizer for heading generation
summarizer = pipeline("summarization", model="facebook/bart-large-cnn", device=0 if torch.cuda.is_available() else -1)
tokenizer = AutoTokenizer.from_pretrained("facebook/bart-large-cnn", use_fast=True)

# Read and clean the transcript
with open(input_file, 'r', encoding='utf-8') as file:
    transcript = file.read()

# Clean special characters and excessive spaces
transcript = re.sub(r'[^\x00-\x7F]+', ' ', transcript)  # Remove non-ASCII characters
transcript = re.sub(r'\s+', ' ', transcript).strip()     # Replace multiple spaces with a single space

# Split transcript into paragraphs based on double newlines or fallback to sentence tokenization
paragraphs = transcript.split('\n\n') if '\n\n' in transcript else [transcript]

# Create clearer and more coherent chunks
chunk_size = 300  # Limit chunk size to ~400 tokens for clarity and coherence
chunks = []

for paragraph in paragraphs:
    sentences = sent_tokenize(paragraph)
    current_chunk = []

    for sentence in sentences:
        current_chunk.append(sentence)
        # Check token length instead of word count for more reliable chunking
        token_count = len(tokenizer(' '.join(current_chunk))['input_ids'])
        if token_count >= chunk_size:
            chunks.append(' '.join(current_chunk))
            current_chunk = []

    # Append remaining sentences in the current chunk
    if current_chunk:
        chunks.append(' '.join(current_chunk))

# Generate headings and organize content
segmented_content = ""
for idx, chunk in enumerate(chunks, 1):
    if not chunk.strip() or len(chunk.split()) < 5:
        continue  # Skip empty or very short chunks

    # Limit chunk to 1024 tokens if it's too large
    tokens = tokenizer(chunk)['input_ids']
    if len(tokens) > 1024:
        chunk = tokenizer.decode(tokens[:1024])

    # Debugging print to track token info
    print(f"Processing chunk {idx}/{len(chunks)} with {len(tokens)} tokens.")

    # Generate a heading using summarization
    heading = summarizer(chunk, max_length=10, min_length=3, do_sample=False)[0]['summary_text']
    segmented_content += f"### {heading.strip()}\n\n{chunk.strip()}\n\n"

# Save segmented content to a new file
with open(output_file, 'w', encoding='utf-8') as file:
    file.write(segmented_content)

print(f"Segmented transcript saved to {output_file}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Device set to use cpu
Processing chunk 1/2 with 2950 tokens.
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-37-f582a8017948> in <cell line: 0>()
     63
     64     # Generate a heading using summarization
---> 65     heading = summarizer(chunk, max_length=10, min_length=3, do_sample=False)[0]['summary_text']
     66     segmented_content += f"### {heading.strip()}\n\n{chunk.strip()}\n\n"
     67

                              16 frames
/usr/local/lib/python3.11/dist-packages/torch/nn/functional.py in embedding(input, weight, padding_idx, max_norm, norm_type,
scale_grad_by_freq, sparse)
   2549         # remove once script supports set_grad_enabled
   2550         _no_grad_embedding_renorm_(weight, input, max_norm, norm_type)
-> 2551     return torch.embedding(weight, input, padding_idx, scale_grad_by_freq, sparse)
   2552
   2553

IndexError: index out of range in self
```

```
import os
import re
import nltk
from transformers import pipeline, AutoTokenizer
from nltk.tokenize import sent_tokenize
import torch

# Download the Punkt tokenizer for sentence splitting if not already downloaded
nltk.download('punkt')

# Define file paths
input_file = '/content/Kalsi sir transcript.txt'  # Replace with your transcript file name
output_file = 'segmented_transcript.txt'

# Load the summarization model and tokenizer for heading generation
summarizer = pipeline("summarization", model="facebook/bart-large-cnn", device=0 if torch.cuda.is_available() else -1)
tokenizer = AutoTokenizer.from_pretrained("facebook/bart-large-cnn", use_fast=True)

# Re-download the tokenizer files to fix potential corruption
tokenizer.save_pretrained("./tokenizer")
tokenizer = AutoTokenizer.from_pretrained("./tokenizer")

# Read and clean the transcript
with open(input_file, 'r', encoding='utf-8') as file:
    transcript = file.read()

# Clean special characters and excessive spaces
transcript = re.sub(r'[^\x00-\x7F]+', ' ', transcript)  # Remove non-ASCII characters
transcript = re.sub(r'\s+', ' ', transcript).strip()     # Replace multiple spaces with a single space

# Split transcript into paragraphs based on double newlines or fallback to sentence tokenization
paragraphs = transcript.split('\n\n') if '\n\n' in transcript else [transcript]

# Create clearer and more coherent chunks
chunk_size = 400  # Limit chunk size to ~400 tokens for clarity and coherence
chunks = []

for paragraph in paragraphs:
    sentences = sent_tokenize(paragraph)
    current_chunk = []

    for sentence in sentences:
        current_chunk.append(sentence)
        # Check token length instead of word count for more reliable chunking
        token_count = len(tokenizer(' '.join(current_chunk))['input_ids'])
        if token_count >= chunk_size:
            chunks.append(' '.join(current_chunk))
            current_chunk = []

    # Append remaining sentences in the current chunk
    if current_chunk:
        chunks.append(' '.join(current_chunk))

# Generate headings and organize content
segmented_content = ""
for idx, chunk in enumerate(chunks, 1):
    if not chunk.strip() or len(chunk.split()) < 5:
        continue  # Skip empty or very short chunks
```

```

    # Tokenize the chunk
    tokens = tokenizer(chunk)['input_ids']

    # Split chunk into smaller parts if it exceeds 1024 tokens
    if len(tokens) > 1024:
        sub_chunks = [tokens[i:i + 1024] for i in range(0, len(tokens), 1024)]
    else:
        sub_chunks = [tokens]

    # Process each sub-chunk separately
    for sub_idx, sub_chunk in enumerate(sub_chunks, 1):
        # Decode sub-chunk back to text
        sub_chunk_text = tokenizer.decode(sub_chunk, skip_special_tokens=True)

        # Debugging print to track token info
        print(f"Processing chunk {idx}.{sub_idx}/{len(chunks)} with {len(sub_chunk)} tokens.")

        # Generate a heading using summarization
        try:
            heading = summarizer(sub_chunk_text, max_length=10, min_length=3, do_sample=False)[0]['summary_text']
            segmented_content += f"### {heading.strip()}\n\n{sub_chunk_text.strip()}\n\n"
        except IndexError as e:
            print(f"Skipping a chunk due to error: {e}")
            continue

# Save segmented content to a new file
with open(output_file, 'w', encoding='utf-8') as file:
    file.write(segmented_content)

print(f"Segmented transcript saved to {output_file}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Device set to use cpu
Processing chunk 1.1/2 with 1024 tokens.
Skipping a chunk due to error: index out of range in self
Processing chunk 1.2/2 with 1024 tokens.
Skipping a chunk due to error: index out of range in self
Processing chunk 1.3/2 with 902 tokens.
Processing chunk 2.1/2 with 1024 tokens.
Skipping a chunk due to error: index out of range in self
Processing chunk 2.2/2 with 1024 tokens.
Skipping a chunk due to error: index out of range in self
Processing chunk 2.3/2 with 1024 tokens.
Skipping a chunk due to error: index out of range in self
Processing chunk 2.4/2 with 482 tokens.
Segmented transcript saved to segmented_transcript.txt
```

```
import os
import re
import nltk
from transformers import pipeline, AutoTokenizer, AutoModel
from sentence_transformers import SentenceTransformer, util
from nltk.tokenize import sent_tokenize
import torch

# Download the Punkt tokenizer for sentence splitting if not already downloaded
nltk.download('punkt')

# Define file paths
input_file = '/content/Kalsi sir transcript.txt'  # Replace with your transcript file name
output_file = 'segmented_transcript.txt'

# Load summarization and zero-shot classification models
summarizer = pipeline("summarization", model="facebook/bart-large-cnn", device=0 if torch.cuda.is_available() else -1)
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
tokenizer = AutoTokenizer.from_pretrained("facebook/bart-large-cnn", use_fast=True)

# Load sentence-transformers model for semantic similarity
embedder = SentenceTransformer('all-MiniLM-L6-v2')

# Read and clean the transcript
with open(input_file, 'r', encoding='utf-8') as file:
    transcript = file.read()

# Clean special characters and excessive spaces
transcript = re.sub(r'[^\x00-\x7F]+', ' ', transcript)  # Remove non-ASCII characters
transcript = re.sub(r'\s+', ' ', transcript).strip()     # Replace multiple spaces with a single space

# Split transcript into sentences
sentences = sent_tokenize(transcript)

```

```
# Create chunks based on semantic similarity and concept shifts
chunk_size = 300
chunks = []
current_chunk = []
current_embedding = None

for sentence in sentences:
    # Embed the current sentence
    sentence_embedding = embedder.encode(sentence, convert_to_tensor=True)

    # Check if this is the first sentence
    if current_embedding is None:
        current_embedding = sentence_embedding
        current_chunk.append(sentence)
        continue

    # Calculate similarity with the last chunk's embedding
    similarity = util.pytorch_cos_sim(current_embedding, sentence_embedding).item()

    # If similarity is low or chunk size exceeds limit, start a new chunk
    if similarity < 0.6 or len(tokenizer(' '.join(current_chunk))['input_ids']) >= chunk_size:
        chunks.append(' '.join(current_chunk))
        current_chunk = [sentence]
        current_embedding = sentence_embedding
    else:
        current_chunk.append(sentence)
        current_embedding = (current_embedding + sentence_embedding) / 2  # Update embedding with average

# Append remaining chunk if exists
if current_chunk:
    chunks.append(' '.join(current_chunk))

# Generate headings and organize content
segmented_content = ""
for idx, chunk in enumerate(chunks, 1):
    if not chunk.strip() or len(chunk.split()) < 5:
        continue  # Skip empty or very short chunks

    # Generate a heading using zero-shot classification for topic detection
    topics = ["Technology", "Education", "AI and Machine Learning", "Research", "Career Guidance", "Software Development", "Leadersh
    classification = classifier(chunk, candidate_labels=topics, multi_label=False)
    main_topic = classification['labels'][0]

    # Generate a concise heading based on main topic
    heading = summarizer(chunk, max_length=15, min_length=5, do_sample=False)[0]['summary_text']
    segmented_content += f"### {main_topic}: {heading.strip()}\n\n{chunk.strip()}\n\n"

# Save segmented content to a new file
with open(output_file, 'w', encoding='utf-8') as file:
    file.write(segmented_content)

print(f"Segmented transcript saved to {output_file}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%                                   1.58k/1.58k [00:00<00:00, 30.1kB/s]
model.safetensors: 100%                             1.63G/1.63G [00:21<00:00, 227MB/s]
generation_config.json: 100%                        363/363 [00:00<00:00, 19.6kB/s]
vocab.json: 100%                                    899k/899k [00:00<00:00, 1.98MB/s]
merges.txt: 100%                                    456k/456k [00:00<00:00, 11.3MB/s]
tokenizer.json: 100%                                1.36M/1.36M [00:00<00:00, 18.9MB/s]
Device set to use cpu
config.json: 100%                                   1.15k/1.15k [00:00<00:00, 72.2kB/s]
model.safetensors: 100%                             1.63G/1.63G [00:09<00:00, 180MB/s]
tokenizer_config.json: 100%                         26.0/26.0 [00:00<00:00, 1.69kB/s]
vocab.json: 100%                                    899k/899k [00:00<00:00, 17.6MB/s]
merges.txt: 100%                                    456k/456k [00:00<00:00, 30.5MB/s]
tokenizer.json: 100%                                1.36M/1.36M [00:00<00:00, 38.5MB/s]
Device set to use cpu
modules.json: 100%                                  349/349 [00:00<00:00, 33.9kB/s]
config_sentence_transformers.json: 100%             116/116 [00:00<00:00, 10.1kB/s]
README.md: 100%                                     10.7k/10.7k [00:00<00:00, 799kB/s]
sentence_bert_config.json: 100%                     53.0/53.0 [00:00<00:00, 3.47kB/s]
config.json: 100%                                   612/612 [00:00<00:00, 42.5kB/s]
model.safetensors: 100%                             90.9M/90.9M [00:00<00:00, 180MB/s]
tokenizer_config.json: 100%                         350/350 [00:00<00:00, 26.8kB/s]
vocab.txt: 100%                                     232k/232k [00:00<00:00, 17.4MB/s]
tokenizer.json: 100%                                466k/466k [00:00<00:00, 25.0MB/s]
special_tokens_map.json: 100%                       112/112 [00:00<00:00, 6.13kB/s]
config.json: 100%                                   190/190 [00:00<00:00, 17.9kB/s]
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-1-8e707ad1f898> in <cell line: 0>()
     31
     32 # Split transcript into sentences
     35 # Create chunks based on semantic similarity and concept shifts
```

```
#Transcript With timestamp
```

```
import re
import urllib.parse
import requests
from youtube_transcript_api import YouTubeTranscriptApi
from pytube import YouTube
import speech_recognition as sr
from pydub import AudioSegment
import os

def extract_video_id(video_url):
    """
    Extracts the YouTube video ID from various URL formats.
    """
    parsed_url = urllib.parse.urlparse(video_url)
    query_params = urllib.parse.parse_qs(parsed_url.query)

    if "v" in query_params:
        return query_params["v"][0]

    match = re.search(r"(youtu\.be/|youtube\.com/embed/|youtube\.com/shorts/)([\w-]+)", video_url)
    if match:
        return match.group(2)

    return None

def download_audio(video_url):
```

```
    """
    Downloads the audio using yt-dlp with cookies and returns the file path.
    """
    try:
        ydl_opts = {
            'format': 'bestaudio/best',
            'outtmpl': 'audio.%(ext)s',
            'cookiefile': 'cookies (1).txt',  # Use the exported cookies
            'postprocessors': [{
                'key': 'FFmpegExtractAudio',
                'preferredcodec': 'mp3',
                'preferredquality': '192',
            }],
        }
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            info = ydl.extract_info(video_url, download=True)
            return "audio.mp3"
    except Exception as e:
        return f"Error downloading audio: {str(e)}"

def convert_audio_to_wav(audio_file):
    """
    Converts the downloaded MP3 audio to WAV format using pydub.
    """
    wav_file = "audio.wav"
    try:
        AudioSegment.from_mp3(audio_file).export(wav_file, format="wav")
        return wav_file
    except Exception as e:
        return f"Error converting to WAV: {str(e)}"

def transcribe_audio(audio_path, chunk_length=30):
    """
    Splits audio into smaller chunks and transcribes each chunk separately.
    Args:
        audio_path (str): Path to the audio file.
        chunk_length (int): Length of each chunk in seconds (default: 30).
    Returns:
        list: List of dictionaries containing transcribed text and timestamps.
    """
    recognizer = sr.Recognizer()
    audio = AudioSegment.from_wav(audio_path)
    total_duration = len(audio) / 1000  # Convert to seconds
    transcribed_segments = []

    print("Transcribing audio in chunks...")

    # Split and transcribe audio in chunks
    for start in range(0, int(total_duration), chunk_length):
        end = min(start + chunk_length, int(total_duration))
        chunk = audio[start * 1000:end * 1000]  # Extract chunk in milliseconds
        chunk.export("chunk.wav", format="wav")  # Save chunk temporarily

        with sr.AudioFile("chunk.wav") as source:
            try:
                audio_data = recognizer.record(source)
                text = recognizer.recognize_google(audio_data)
                transcribed_segments.append({
                    "start": start,
                    "end": end,
                    "text": text
                })
            except sr.UnknownValueError:
                transcribed_segments.append({
                    "start": start,
                    "end": end,
                    "text": "[Unintelligible]"
                })
            except sr.RequestError as e:
                return f"Error with the speech recognition service: {str(e)}"

    os.remove("chunk.wav")  # Clean up temporary chunk file
    return transcribed_segments

def get_transcript_unlisted(video_url):
    """
    Tries to fetch the transcript using youtube_transcript_api first,
    then falls back to downloading and transcribing audio if necessary.
    """
    video_id = extract_video_id(video_url)
    if not video_id:
        return "Invalid YouTube URL."
```

```python

    # Try to fetch transcript using youtube_transcript_api
    try:
        transcript = YouTubeTranscriptApi.get_transcript(video_id)
        # Add 'end' time to each segment
        for segment in transcript:
            segment["end"] = segment["start"] + segment["duration"]
        return transcript  # Return transcript with timestamps
    except:
        print("Transcript not available via API, attempting audio transcription...")

    # Download and transcribe audio if no transcript is available
    audio_file = download_audio(video_url)
    if "Error" in audio_file:
        return audio_file

    wav_file = convert_audio_to_wav(audio_file)
    if "Error" in wav_file:
        return wav_file

    transcription = transcribe_audio(wav_file)

    # Cleanup temporary files
    os.remove(audio_file)
    os.remove(wav_file)

    return transcription

# Example usage
if __name__ == "__main__":
    video_url = input("Enter the YouTube video URL: ")
    transcript = get_transcript_unlisted(video_url)

    if isinstance(transcript, list):
        print("\nTranscript with Timestamps:")
        for segment in transcript:
            print(f"{segment['start']} - {segment['end']}: {segment['text']}")
    else:
        print("\nTranscript:\n", transcript)
```

```
Enter the YouTube video URL: https://youtu.be/sK8SILOM37I

Transcript with Timestamps:
0.08 - 5.92: so sir we know that India has seen a
3.04 - 8.280000000000001: huge Revolution with digital payments we
5.92 - 9.679: all thought that India is a place at
8.28 - 12.559: least the West thought that India is a
9.679 - 14.799: place where uh many people do not get a
12.559 - 17.039: square meal right that was the narrative
14.799 - 21.358999999999998: some 30 years ago and not many are
17.039 - 24.840000000000003: literate people cannot read um but then
21.359 - 26.679000000000002: we have now shown that digital payments
24.84 - 29.32: number one is India while people thought
26.679 - 30.759: that it wouldn't even come to top 50 let
29.32 - 32.88: alone number one
30.759 - 34.16: I think immediately after UPA the next
32.88 - 37.84: big
34.16 - 40.599999999999994: revolution personally I think is in
37.84 - 43.28: education and the complete homework for
40.6 - 45.68: this has happened in the form of NEP the
43.28 - 48.079: documentation of which many of us have
45.68 - 52.16: read and realize that it is too good to
48.079 - 53.64: be true followed by which we got ncrf
52.16 - 55.68: framework done which is the national
53.64 - 57.44: credit framework surrounding which we'll
55.68 - 60.48: be discussing
57.44 - 63.76: today sir my question is do you think
60.48 - 67.15899999999999: ncrf plus NEP put together will be the
63.76 - 69.2: next big revolution after UPA in
67.159 - 72.0: India
69.2 - 77.36: absolutely and why I think
72.0 - 80.119: so is because in education the last
77.36 - 82.56: policy came up many many years ago that
80.119 - 87.079: was in
82.56 - 87.07900000000001: 1986 which was slightly tweaked in
87.72 - 93.2: 1982 and thereafter so so many changes
90.68 - 96.399: have happened in the real
93.2 - 99.119: world so many changes have happened in
96.399 - 102.32000000000001: the requirement of the
99.119 - 104.2: industry requirement of Manpower for the
102.32 - 106.19999999999999: industry however there were no
104.2 - 107.479: corresponding changes which happened
106.2 - 111.0: education
107.479 - 113.32: system so therefore I feel that this was
111.0 - 116.52: the right time when we brought
113.32 - 121.79899999999999: in the education policy 2020 honorable
```

```
116.52 - 125.19999999999999: prime minister declared it open in July
121.799 - 129.399: 2020 and we recently celebrated the
125.2 - 133.4: fourth anniversary of NEP 2020 ncrf has
129.399 - 136.8: been brought to implement the intent of
133.4 - 139.36: 2020 2020 is a policy the NP 2020 is a
136.8 - 141.44: policy and for implementing a policy you
139.36 - 144.48000000000002: need a
141.44 - 149.04: framework now why we call it a framework
144.48 - 151.16: we call it a framework because this is
149.04 - 154.35999999999999: very flexible
151.16 - 159.4: this allows you all the
```

```
pip install youtube_transcript_api
```

```
Collecting youtube_transcript_api
  Downloading youtube_transcript_api-0.6.3-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: defusedxml<0.8.0,>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from youtube_transcript_api) (0
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from youtube_transcript_api) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcrip
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api) (3.16
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api
Downloading youtube_transcript_api-0.6.3-py3-none-any.whl (622 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 622.3/622.3 kB 25.5 MB/s eta 0:00:00
Installing collected packages: youtube_transcript_api
Successfully installed youtube_transcript_api-0.6.3
```

```
pip install yt-dlp
```

```
Collecting yt-dlp
  Downloading yt_dlp-2025.2.19-py3-none-any.whl.metadata (171 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 171.9/171.9 kB 9.7 MB/s eta 0:00:00
Downloading yt_dlp-2025.2.19-py3-none-any.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 47.6 MB/s eta 0:00:00
Installing collected packages: yt-dlp
Successfully installed yt-dlp-2025.2.19
```

```
pip install pytube
```

```
Collecting pytube
  Downloading pytube-15.0.0-py3-none-any.whl.metadata (5.0 kB)
Downloading pytube-15.0.0-py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.6/57.6 kB 4.1 MB/s eta 0:00:00
Installing collected packages: pytube
Successfully installed pytube-15.0.0
```

```
pip install SpeechRecognition
```

```
Collecting SpeechRecognition
  Downloading SpeechRecognition-3.14.1-py3-none-any.whl.metadata (31 kB)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from SpeechRecognition) (4.12.2)
Downloading SpeechRecognition-3.14.1-py3-none-any.whl (32.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 32.9/32.9 MB 17.4 MB/s eta 0:00:00
Installing collected packages: SpeechRecognition
Successfully installed SpeechRecognition-3.14.1
```

```
pip install pydub
```

```
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub
Successfully installed pydub-0.25.1
```

```
import re
import urllib.parse
import requests
from youtube_transcript_api import YouTubeTranscriptApi
from pytube import YouTube
import speech_recognition as sr
from pydub import AudioSegment
import os

def extract_video_id(video_url):
    """
    Extracts the YouTube video ID from various URL formats.
    """
    parsed_url = urllib.parse.urlparse(video_url)
    query_params = urllib.parse.parse_qs(parsed_url.query)

    if "v" in query_params:
```

```
        return query_params["v"][0]

    match = re.search(r"(youtu\.be/|youtube\.com/embed/|youtube\.com/shorts/)([\w-]+)", video_url)
    if match:
        return match.group(2)

    return None

def download_audio(video_url):
    """
    Downloads the audio using yt-dlp with cookies and returns the file path.
    """
    try:
        ydl_opts = {
            'format': 'bestaudio/best',
            'outtmpl': 'audio.%(ext)s',
            'cookiefile': 'cookies (1).txt',  # Use the exported cookies
            'postprocessors': [{
                'key': 'FFmpegExtractAudio',
                'preferredcodec': 'mp3',
                'preferredquality': '192',
            }],
        }
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            info = ydl.extract_info(video_url, download=True)
            return "audio.mp3"
    except Exception as e:
        return f"Error downloading audio: {str(e)}"

def convert_audio_to_wav(audio_file):
    """
    Converts the downloaded MP3 audio to WAV format using pydub.
    """
    wav_file = "audio.wav"
    try:
        AudioSegment.from_mp3(audio_file).export(wav_file, format="wav")
        return wav_file
    except Exception as e:
        return f"Error converting to WAV: {str(e)}"

def transcribe_audio(audio_path, chunk_length=30):
    """
    Splits audio into smaller chunks and transcribes each chunk separately.
    Args:
        audio_path (str): Path to the audio file.
        chunk_length (int): Length of each chunk in seconds (default: 30).
    Returns:
        list: List of dictionaries containing transcribed text and timestamps.
    """
    recognizer = sr.Recognizer()
    audio = AudioSegment.from_wav(audio_path)
    total_duration = len(audio) / 1000  # Convert to seconds
    transcribed_segments = []

    print("Transcribing audio in chunks...")

    # Split and transcribe audio in chunks
    for start in range(0, int(total_duration), chunk_length):
        end = min(start + chunk_length, int(total_duration))
        chunk = audio[start * 1000:end * 1000]  # Extract chunk in milliseconds
        chunk.export("chunk.wav", format="wav")  # Save chunk temporarily

        with sr.AudioFile("chunk.wav") as source:
            try:
                audio_data = recognizer.record(source)
                text = recognizer.recognize_google(audio_data)
                transcribed_segments.append({
                    "start": start,
                    "end": end,
                    "text": text
                })
            except sr.UnknownValueError:
                transcribed_segments.append({
                    "start": start,
                    "end": end,
                    "text": "[Unintelligible]"
                })
            except sr.RequestError as e:
                return f"Error with the speech recognition service: {str(e)}"

    os.remove("chunk.wav")  # Clean up temporary chunk file
    return transcribed_segments
```

```python

def get_transcript_unlisted(video_url):
    """
    Tries to fetch the transcript using youtube_transcript_api first,
    then falls back to downloading and transcribing audio if necessary.
    """
    video_id = extract_video_id(video_url)
    if not video_id:
        return "Invalid YouTube URL."

    # Try to fetch transcript using youtube_transcript_api
    try:
        transcript = YouTubeTranscriptApi.get_transcript(video_id)
        # Add 'end' time to each segment
        for segment in transcript:
            segment["end"] = segment["start"] + segment["duration"]
        return transcript  # Return transcript with timestamps
    except:
        print("Transcript not available via API, attempting audio transcription...")

    # Download and transcribe audio if no transcript is available
    audio_file = download_audio(video_url)
    if "Error" in audio_file:
        return audio_file

    wav_file = convert_audio_to_wav(audio_file)
    if "Error" in wav_file:
        return wav_file

    transcription = transcribe_audio(wav_file)

    # Cleanup temporary files
    os.remove(audio_file)
    os.remove(wav_file)

    return transcription

def save_transcript_to_file(transcript, filename="transcript.txt"):
    """
    Saves the transcript to a text file.
    Args:
        transcript (list or str): The transcript to save.
        filename (str): The name of the output file.
    """
    with open(filename, "w", encoding="utf-8") as file:
        if isinstance(transcript, list):
            for segment in transcript:
                file.write(f"{segment['start']} - {segment['end']}: {segment['text']}\n")
        else:
            file.write(transcript)
    print(f"Transcript saved to {filename}")

# Example usage
if __name__ == "__main__":
    video_url = input("Enter the YouTube video URL: ")
    transcript = get_transcript_unlisted(video_url)

    if isinstance(transcript, list):
        print("\nTranscript with Timestamps:")
        for segment in transcript:
            print(f"{segment['start']} - {segment['end']}: {segment['text']}")
    else:
        print("\nTranscript:\n", transcript)

    # Save transcript to a text file
    save_transcript_to_file(transcript, "transcript.txt")
```

```
Enter the YouTube video URL: https://youtu.be/sK8SILOM37I

Transcript with Timestamps:
0.08 - 5.92: so sir we know that India has seen a
3.04 - 8.280000000000001: huge Revolution with digital payments we
5.92 - 9.679: all thought that India is a place at
8.28 - 12.559: least the West thought that India is a
9.679 - 14.799: place where uh many people do not get a
12.559 - 17.039: square meal right that was the narrative
14.799 - 21.358999999999998: some 30 years ago and not many are
17.039 - 24.840000000000003: literate people cannot read um but then
21.359 - 26.679000000000002: we have now shown that digital payments
24.84 - 29.32: number one is India while people thought
26.679 - 30.759: that it wouldn't even come to top 50 let
29.32 - 32.88: alone number one
30.759 - 34.16: I think immediately after UPA the next
32.88 - 37.84: big
```

```
34.16 - 40.599999999999994: revolution personally I think is in
37.84 - 43.28: education and the complete homework for
40.6 - 45.68: this has happened in the form of NEP the
43.28 - 48.079: documentation of which many of us have
45.68 - 52.16: read and realize that it is too good to
48.079 - 53.64: be true followed by which we got ncrf
52.16 - 55.68: framework done which is the national
53.64 - 57.44: credit framework surrounding which we'll
55.68 - 60.48: be discussing
57.44 - 63.76: today sir my question is do you think
60.48 - 67.15899999999999: ncrf plus NEP put together will be the
63.76 - 69.2: next big revolution after UPA in
67.159 - 72.0: India
69.2 - 77.36: absolutely and why I think
72.0 - 80.119: so is because in education the last
77.36 - 82.56: policy came up many many years ago that
80.119 - 87.079: was in
82.56 - 87.07900000000001: 1986 which was slightly tweaked in
87.72 - 93.2: 1982 and thereafter so so many changes
90.68 - 96.399: have happened in the real
93.2 - 99.119: world so many changes have happened in
96.399 - 102.32000000000001: the requirement of the
99.119 - 104.2: industry requirement of Manpower for the
102.32 - 106.19999999999999: industry however there were no
104.2 - 107.479: corresponding changes which happened
106.2 - 111.0: education
107.479 - 113.32: system so therefore I feel that this was
111.0 - 116.52: the right time when we brought
113.32 - 121.79899999999999: in the education policy 2020 honorable
116.52 - 125.19999999999999: prime minister declared it open in July
121.799 - 129.399: 2020 and we recently celebrated the
125.2 - 133.4: fourth anniversary of NEP 2020 ncrf has
129.399 - 136.8: been brought to implement the intent of
133.4 - 139.36: 2020 2020 is a policy the NP 2020 is a
136.8 - 141.44: policy and for implementing a policy you
139.36 - 144.48000000000002: need a
141.44 - 149.04: framework now why we call it a framework
144.48 - 151.16: we call it a framework because this is
149.04 - 154.35999999999999: very flexible
151.16 - 159.4: this allows you all the
154.36 - 161.84: Innovation the way you educate your kids
```

```
import json
from sklearn.cluster import KMeans
from sentence_transformers import SentenceTransformer
import numpy as np

def load_transcript_from_file(file_path):
    """
    Loads the transcript from a file.
    Args:
        file_path (str): Path to the transcript file.
    Returns:
        list: List of dictionaries with 'start', 'end', and 'text' keys.
    """
    try:
        with open(file_path, "r", encoding="utf-8") as file:
            if file_path.endswith(".json"):
                transcript = json.load(file)
            else:
                # For plain text files, assume each line is in the format: [start] - [end]: [text]
                transcript = []
                for line in file:
                    match = re.match(r"(\d+\.\d+) - (\d+\.\d+): (.+)", line.strip())
                    if match:
                        start, end, text = match.groups()
                        transcript.append({
                            "start": float(start),
                            "end": float(end),
                            "text": text
                        })
            return transcript
    except Exception as e:
        print(f"Error loading transcript: {e}")
        return None

def segment_transcript_semantically(transcript, num_clusters=5):
    """
    Segments the transcript into semantically meaningful clusters.
    Args:
        transcript (list): List of dictionaries with 'start', 'end', and 'text' keys.
        num_clusters (int): Number of clusters to create (default: 5).
    Returns:
        list: List of dictionaries with 'start', 'end', and 'text' for each segment.
    """
    # Extract sentences and their timestamps
```

```
    sentences = [segment["text"] for segment in transcript]
    timestamps = [(segment["start"], segment["end"]) for segment in transcript]

    # Load a pre-trained sentence embedding model
    model = SentenceTransformer('all-MiniLM-L6-v2')
    sentence_embeddings = model.encode(sentences)

    # Perform K-Means clustering
    kmeans = KMeans(n_clusters=num_clusters, random_state=42)
    clusters = kmeans.fit_predict(sentence_embeddings)

    # Group sentences by cluster
    clustered_segments = {}
    for i, cluster in enumerate(clusters):
        if cluster not in clustered_segments:
            clustered_segments[cluster] = []
        clustered_segments[cluster].append({
            "start": timestamps[i][0],
            "end": timestamps[i][1],
            "text": sentences[i]
        })

    # Merge segments in each cluster
    segmented_transcript = []
    for cluster, segments in clustered_segments.items():
        start_time = segments[0]["start"]
        end_time = segments[-1]["end"]
        combined_text = " ".join([segment["text"] for segment in segments])
        segmented_transcript.append({
            "start": start_time,
            "end": end_time,
            "text": combined_text
        })

    # Sort segments by start time
    segmented_transcript.sort(key=lambda x: x["start"])

    return segmented_transcript

def save_segmented_transcript(segmented_transcript, output_path="segmented_transcript.txt"):
    """
    Saves the segmented transcript to a text file.
    Args:
        segmented_transcript (list): List of dictionaries with 'start', 'end', and 'text'.
        output_path (str): The path to save the output file.
    """
    with open(output_path, "w", encoding="utf-8") as file:
        for segment in segmented_transcript:
            file.write(f"{segment['start']} - {segment['end']}: {segment['text']}\n")
    print(f"Segmented transcript saved to {output_path}")

# Example usage
if __name__ == "__main__":
    # Ask the user for the transcript file path
    transcript_path = input("Enter the path to the transcript file: ")
    output_path = input("Enter the path to save the segmented transcript (default: segmented_transcript.txt): ") or "segmented_tra

    # Load the transcript from the file
    transcript = load_transcript_from_file(transcript_path)
    if not transcript:
        print("Failed to load transcript. Exiting.")
        exit()

    # Segment the transcript semantically
    segmented_transcript = segment_transcript_semantically(transcript, num_clusters=3)

    # Print the segmented transcript
    print("\nSegmented Transcript:")
    for segment in segmented_transcript:
        print(f"{segment['start']} - {segment['end']}: {segment['text']}")

    # Save the segmented transcript to a file
    save_segmented_transcript(segmented_transcript, output_path)
```

```
Enter the path to the transcript file: /content/transcript.txt
Enter the path to save the segmented transcript (default: segmented_transcript.txt):

Segmented Transcript:
0.08 - 3444.2: so sir we know that India has seen a all thought that India is a place at least the West thought that India is a plac
3.04 - 3436.7599999999998: huge Revolution with digital payments we documentation of which many of us have framework done which is t
37.84 - 3444.2000000000003: education and the complete homework for so is because in education the last education in the education p
Segmented transcript saved to segmented_transcript.txt
```

```python
import json
import re
from bertopic import BERTopic
from sentence_transformers import SentenceTransformer
from sklearn.feature_extraction.text import CountVectorizer

def load_transcript_from_file(file_path):
    """
    Loads the transcript from a file.
    Args:
        file_path (str): Path to the transcript file.
    Returns:
        list: List of dictionaries with 'start', 'end', and 'text' keys.
    """
    try:
        with open(file_path, "r", encoding="utf-8") as file:
            if file_path.endswith(".json"):
                transcript = json.load(file)
            else:
                # For plain text files, assume each line is in the format: [start] - [end]: [text]
                transcript = []
                for line in file:
                    match = re.match(r"(\d+\.\d+) - (\d+\.\d+): (.+)", line.strip())
                    if match:
                        start, end, text = match.groups()
                        transcript.append({
                            "start": float(start),
                            "end": float(end),
                            "text": text
                        })
            return transcript
    except Exception as e:
        print(f"Error loading transcript: {e}")
        return None

def segment_transcript_with_topics(transcript, num_topics=5):
    """
    Segments the transcript into topics using BERTopic.
    Args:
        transcript (list): List of dictionaries with 'start', 'end', and 'text' keys.
        num_topics (int): Number of topics to create (default: 5).
    Returns:
        list: List of dictionaries with 'start', 'end', 'text', and 'topic' for each segment.
    """
    # Extract sentences and their timestamps
    sentences = [segment["text"] for segment in transcript]
    timestamps = [(segment["start"], segment["end"]) for segment in transcript]

    # Load a pre-trained sentence embedding model
    embedding_model = SentenceTransformer('all-MiniLM-L6-v2')

    # Initialize BERTopic
    vectorizer_model = CountVectorizer(stop_words="english")  # Remove stopwords for better topic modeling
    topic_model = BERTopic(embedding_model=embedding_model, vectorizer_model=vectorizer_model, nr_topics=num_topics)

    # Fit the model and transform the sentences
    topics, _ = topic_model.fit_transform(sentences)

    # Assign topics to each sentence
    segmented_transcript = []
    for i, (start, end) in enumerate(timestamps):
        segmented_transcript.append({
            "start": start,
            "end": end,
            "text": sentences[i],
            "topic": int(topics[i])
        })

    return segmented_transcript, topic_model

def save_segmented_transcript(segmented_transcript, topic_model, output_path="segmented_transcript.txt"):
    """
    Saves the segmented transcript with topics to a text file.
    Args:
        segmented_transcript (list): List of dictionaries with 'start', 'end', 'text', and 'topic'.
        topic_model (BERTopic): The trained BERTopic model.
        output_path (str): The path to save the output file.
    """
    # Get topic labels
```

```
    topic_info = topic_model.get_topic_info()
    topic_labels = {row["Topic"]: row["Name"] for _, row in topic_info.iterrows()}

    with open(output_path, "w", encoding="utf-8") as file:
        for segment in segmented_transcript:
            topic_label = topic_labels.get(segment["topic"], "Unknown Topic")
            file.write(f"{segment['start']} - {segment['end']} [{topic_label}]: {segment['text']}\n")
    print(f"Segmented transcript saved to {output_path}")

# Example usage
if __name__ == "__main__":
    # Ask the user for the transcript file path
    transcript_path = input("Enter the path to the transcript file: ")
    output_path = input("Enter the path to save the segmented transcript (default: segmented_transcript.txt): ") or "segmented_tra

    # Load the transcript from the file
    transcript = load_transcript_from_file(transcript_path)
    if not transcript:
        print("Failed to load transcript. Exiting.")
        exit()

    # Segment the transcript into topics
    segmented_transcript, topic_model = segment_transcript_with_topics(transcript, num_topics=5)

    # Print the segmented transcript
    print("\nSegmented Transcript with Topics:")
    for segment in segmented_transcript:
        print(f"{segment['start']} - {segment['end']} [Topic {segment['topic']}]: {segment['text']}")

    # Save the segmented transcript to a file
    save_segmented_transcript(segmented_transcript, topic_model, output_path)
```

```
Enter the path to the transcript file: /content/transcript.txt
Enter the path to save the segmented transcript (default: segmented_transcript.txt):

Segmented Transcript with Topics:
0.08 - 5.92 [Topic 1]: so sir we know that India has seen a
3.04 - 8.280000000000001 [Topic -1]: huge Revolution with digital payments we
5.92 - 9.679 [Topic 1]: all thought that India is a place at
8.28 - 12.559 [Topic 1]: least the West thought that India is a
9.679 - 14.799 [Topic -1]: place where uh many people do not get a
12.559 - 17.039 [Topic -1]: square meal right that was the narrative
14.799 - 21.358999999999998 [Topic -1]: some 30 years ago and not many are
17.039 - 24.840000000000003 [Topic -1]: literate people cannot read um but then
21.359 - 26.679000000000002 [Topic -1]: we have now shown that digital payments
24.84 - 29.32 [Topic 1]: number one is India while people thought
26.679 - 30.759 [Topic 1]: that it wouldn't even come to top 50 let
29.32 - 32.88 [Topic 1]: alone number one
30.759 - 34.16 [Topic 1]: I think immediately after UPA the next
32.88 - 37.84 [Topic 1]: big
34.16 - 40.599999999999994 [Topic 1]: revolution personally I think is in
37.84 - 43.28 [Topic 0]: education and the complete homework for
40.6 - 45.68 [Topic 2]: this has happened in the form of NEP the
43.28 - 48.079 [Topic -1]: documentation of which many of us have
45.68 - 52.16 [Topic -1]: read and realize that it is too good to
48.079 - 53.64 [Topic 2]: be true followed by which we got ncrf
52.16 - 55.68 [Topic 2]: framework done which is the national
53.64 - 57.44 [Topic 0]: credit framework surrounding which we'll
55.68 - 60.48 [Topic -1]: be discussing
57.44 - 63.76 [Topic 1]: today sir my question is do you think
60.48 - 67.15899999999999 [Topic 2]: ncrf plus NEP put together will be the
63.76 - 69.2 [Topic 1]: next big revolution after UPA in
67.159 - 72.0 [Topic 1]: India
69.2 - 77.36 [Topic 1]: absolutely and why I think
72.0 - 80.119 [Topic 0]: so is because in education the last
77.36 - 82.56 [Topic -1]: policy came up many many years ago that
80.119 - 87.079 [Topic -1]: was in
82.56 - 87.07900000000001 [Topic 1]: 1986 which was slightly tweaked in
87.72 - 93.2 [Topic 1]: 1982 and thereafter so so many changes
90.68 - 96.399 [Topic 1]: have happened in the real
93.2 - 99.119 [Topic 1]: world so many changes have happened in
96.399 - 102.32000000000001 [Topic -1]: the requirement of the
99.119 - 104.2 [Topic 0]: industry requirement of Manpower for the
102.32 - 106.19999999999999 [Topic 0]: industry however there were no
104.2 - 107.479 [Topic 1]: corresponding changes which happened
106.2 - 111.0 [Topic 0]: education
107.479 - 113.32 [Topic -1]: system so therefore I feel that this was
111.0 - 116.52 [Topic 1]: the right time when we brought
113.32 - 121.79899999999999 [Topic 0]: in the education policy 2020 honorable
116.52 - 125.19999999999999 [Topic -1]: prime minister declared it open in July
121.799 - 129.399 [Topic -1]: 2020 and we recently celebrated the
125.2 - 133.4 [Topic 2]: fourth anniversary of NEP 2020 ncrf has
129.399 - 136.8 [Topic 0]: been brought to implement the intent of
133.4 - 139.36 [Topic -1]: 2020 2020 is a policy the NP 2020 is a
136.8 - 141.44 [Topic 0]: policy and for implementing a policy you
139.36 - 144.48000000000002 [Topic 1]: need a
141.44 - 149.04 [Topic 2]: framework now why we call it a framework
```

```
144.48 - 151.16 [Topic 2]: we call it a framework because this is
149.04 - 154.35999999999999 [Topic -1]: very flexible
151.16 - 159.4 [Topic 1]: this allows you all the
```

```
pip install bertopic
```

```
Collecting bertopic
  Downloading bertopic-0.16.4-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: hdbscan>=0.8.29 in /usr/local/lib/python3.11/dist-packages (from bertopic) (0.8.40)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.11/dist-packages (from bertopic) (1.26.4)
Requirement already satisfied: pandas>=1.1.5 in /usr/local/lib/python3.11/dist-packages (from bertopic) (2.2.2)
Requirement already satisfied: plotly>=4.7.0 in /usr/local/lib/python3.11/dist-packages (from bertopic) (5.24.1)
Requirement already satisfied: scikit-learn>=0.22.2.post1 in /usr/local/lib/python3.11/dist-packages (from bertopic) (1.6.1)
Requirement already satisfied: sentence-transformers>=0.4.1 in /usr/local/lib/python3.11/dist-packages (from bertopic) (3.4.1)
Requirement already satisfied: tqdm>=4.41.1 in /usr/local/lib/python3.11/dist-packages (from bertopic) (4.67.1)
Requirement already satisfied: umap-learn>=0.5.0 in /usr/local/lib/python3.11/dist-packages (from bertopic) (0.5.7)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.11/dist-packages (from hdbscan>=0.8.29->bertopic) (1.13.1)
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.11/dist-packages (from hdbscan>=0.8.29->bertopic) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.5->bertopic)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.5->bertopic) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.5->bertopic) (2025.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly>=4.7.0->bertopic) (9.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly>=4.7.0->bertopic) (24.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.22.2.post1-
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers>=0.4.1->berto
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers>=0
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers>=0.4.1->bertopic) (1
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.11/dist-packages (from umap-learn>=0.5.0->bertopic) (0.61
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.11/dist-packages (from umap-learn>=0.5.0->bertopic) (0
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transf
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentenc
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-tra
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transf
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20
Requirement already satisfied: llvmlite<0.45,>=0.44.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=0.51.2->umap-le
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.1.5->b
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers>=0
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers>=0.4
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentenc
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformer
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformer
```

```
from sentence_transformers import SentenceTransformer
from sklearn.cluster import KMeans
import re

# Load pre-trained model for sentence embeddings
model = SentenceTransformer('all-MiniLM-L6-v2')

# Function to extract sentences and starting timestamps
def extract_sentences_and_timestamps(file_path):
    sentences = []
    timestamps = []
    # Enhanced regex pattern for range-based timestamps
    pattern = re.compile(r"(\d{1,2}(?:\.\d+)?)\s*-\s*(\d{1,2}(?:\.\d+)?):\s*(.*)")

    try:
        with open(file_path, 'r') as file:
            for line in file:
                line = line.strip()
                if line:
                    match = pattern.match(line)
                    if match:
                        start_time = float(match.group(1))
```

```
                        # Convert start time to hh:mm:ss format
                        hours = int(start_time // 3600)
                        minutes = int((start_time % 3600) // 60)
                        seconds = start_time % 60
                        timestamp = f"{hours:02}:{minutes:02}:{seconds:05.2f}"
                        timestamps.append(timestamp)
                        sentences.append(match.group(3).strip())
                    else:
                        print(f"⚠️ Unmatched line format: {line}")  # Debug print for unmatched lines
    except FileNotFoundError:
        print(f"❌ Error: The file '{file_path}' was not found.")
        return [], []

    if not sentences:
        print("⚠️ Warning: No sentences found. Check if the file is empty or if the format is incorrect.")
    return sentences, timestamps

# Extract sentences and timestamps
input_file = 'transcript.txt'
sentences, timestamps = extract_sentences_and_timestamps(input_file)

# Debug: Print extracted sentences and timestamps
print("\nExtracted Sentences:", sentences)
print("Extracted Timestamps:", timestamps)

# Check if sentences are empty
if not sentences:
    print("❌ Error: No valid sentences were extracted. Exiting the script.")
else:
    # Generate sentence embeddings
    embeddings = model.encode(sentences)

    # Debug: Check if embeddings are generated correctly
    if embeddings.size == 0:
        print("❌ Error: No embeddings were generated. Check if the input text is valid.")
    else:
        # Determine number of clusters (segments)
        num_clusters = 3  # You can adjust this based on your transcript length
        kmeans = KMeans(n_clusters=num_clusters, random_state=0)
        labels = kmeans.fit_predict(embeddings)

        # Create segmented output with timestamps
        segments = [[] for _ in range(num_clusters)]
        for i, label in enumerate(labels):
            segments[label].append(f"[{timestamps[i]}] {sentences[i]}")

        # Save segmented output to a new file
        output_file = 'segmented_transcript.txt'
        with open(output_file, 'w') as file:
            for idx, segment in enumerate(segments):
                file.write(f"--- Segment {idx + 1} ---\n")
                file.write("\n".join(segment))
                file.write("\n\n")

        print(f"✅ Segmented transcript saved to {output_file}")
```

```
⚠️ Unmatched line format: 96.399 - 102.32000000000001: the requirement of the
⚠️ Unmatched line format: 99.119 - 104.2: industry requirement of Manpower for the
⚠️ Unmatched line format: 102.32 - 106.19999999999999: industry however there were no
⚠️ Unmatched line format: 104.2 - 107.479: corresponding changes which happened
⚠️ Unmatched line format: 106.2 - 111.0: education
⚠️ Unmatched line format: 107.479 - 113.32: system so therefore I feel that this was
⚠️ Unmatched line format: 111.0 - 116.52: the right time when we brought
⚠️ Unmatched line format: 113.32 - 121.79899999999999: in the education policy 2020 honorable
⚠️ Unmatched line format: 116.52 - 125.19999999999999: prime minister declared it open in July
⚠️ Unmatched line format: 121.799 - 129.399: 2020 and we recently celebrated the
⚠️ Unmatched line format: 125.2 - 133.4: fourth anniversary of NEP 2020 ncrf has
⚠️ Unmatched line format: 129.399 - 136.8: been brought to implement the intent of
⚠️ Unmatched line format: 133.4 - 139.36: 2020 2020 is a policy the NP 2020 is a
⚠️ Unmatched line format: 136.8 - 141.44: policy and for implementing a policy you
⚠️ Unmatched line format: 139.36 - 144.48000000000002: need a
⚠️ Unmatched line format: 141.44 - 149.04: framework now why we call it a framework
⚠️ Unmatched line format: 144.48 - 151.16: we call it a framework because this is
⚠️ Unmatched line format: 149.04 - 154.35999999999999: very flexible
⚠️ Unmatched line format: 151.16 - 159.4: this allows you all the
⚠️ Unmatched line format: 154.36 - 161.84: Innovation the way you educate your kids
⚠️ Unmatched line format: 159.4 - 165.44: you educate your
⚠️ Unmatched line format: 161.84 - 168.48: students still it provides you the basic
⚠️ Unmatched line format: 165.44 - 171.959: guidelines the the framework the outer
⚠️ Unmatched line format: 168.48 - 174.84: layer it provides and that layer is
⚠️ Unmatched line format: 171.959 - 177.84: mostly the enabling
⚠️ Unmatched line format: 174.84 - 180.76: layer that is such an enabling layer
⚠️ Unmatched line format: 177.84 - 183.48: that it has
⚠️ Unmatched line format: 180.76 - 185.879: broken the shackles which were there in
⚠️ Unmatched line format: 183.48 - 189.48: the education
```

```
⚠️ Unmatched line format: 185.879 - 191.959: sector right so yes it is a big
⚠️ Unmatched line format: 189.48 - 195.23899999999998: Revolution and this is going to change
⚠️ Unmatched line format: 191.959 - 197.68: the way we have been educating our kids
⚠️ Unmatched line format: 195.239 - 201.08: and this will be Game Changer uh for
⚠️ Unmatched line format: 197.68 - 204.56: rest of the world as well so sir I think
⚠️ Unmatched line format: 201.08 - 206.36: um uh let's go with this example of
⚠️ Unmatched line format: 204.56 - 209.799: let's say I started off living in a
⚠️ Unmatched line format: 206.36 - 212.04000000000002: small 2bhk apartment M and I slowly
⚠️ Unmatched line format: 209.799 - 214.15900000000002: developed my kitchen to add to my taste
⚠️ Unmatched line format: 212.04 - 216.76: buts in a way that it's convenient for
⚠️ Unmatched line format: 214.159 - 219.39999999999998: me right and one fine day you came and
⚠️ Unmatched line format: 216.76 - 221.879: you changed my kitchen completely I was
⚠️ Unmatched line format: 219.4 - 226.0: using a bicycle I moved to a scooter and
⚠️ Unmatched line format: 221.879 - 228.2: a car and now you're asking me to fly
⚠️ Unmatched line format: 226.0 - 230.879: and you're giving me an aircraft NEP
⚠️ Unmatched line format: 228.2 - 232.83999999999997: sounds more like that for me how do I do
⚠️ Unmatched line format: 230.879 - 234.92: the transition I fear that I will I will
⚠️ Unmatched line format: 232.84 - 236.239: crash if I use a aeroplane without
⚠️ Unmatched line format: 234.92 - 238.35999999999999: training I'm talking about all the
⚠️ Unmatched line format: 236.239 - 241.12: teachers in the in the country all the
⚠️ Unmatched line format: 238.36 - 244.0: schools in the country we have been
⚠️ Unmatched line format: 241.12 - 245.59900000000002: driving buses at Max now we should fly
⚠️ Unmatched line format: 244.0 - 250.0: how do we do this
⚠️ Unmatched line format: 245.599 - 252.23899999999998: okay look at the requirement of the
⚠️ Unmatched line format: 250.0 - 253.72: industry requirement of the industry has
⚠️ Unmatched line format: 252.239 - 258.4: been moving very
⚠️ Unmatched line format: 253.72 - 262.32: fast the technology is emerging every
⚠️ Unmatched line format: 258.4 - 265.03999999999996: day and the industry is moving with that
```

```
from sentence_transformers import SentenceTransformer
from sklearn.cluster import KMeans
import re

# Load pre-trained model for sentence embeddings
model = SentenceTransformer('all-MiniLM-L6-v2')

# Function to extract sentences and starting timestamps
def extract_sentences_and_timestamps(file_path):
    sentences = []
    timestamps = []
    # Enhanced regex pattern for range-based timestamps
    pattern = re.compile(r"(\d{1,2}(?:\.\d+)?)\s*-\s*(\d{1,2}(?:\.\d+)?):\s*(.*)")

    try:
        with open(file_path, 'r') as file:
            for line in file:
                line = line.strip()
                if line:
                    match = pattern.match(line)
                    if match:
                        start_time = float(match.group(1))
                        # Convert start time to hh:mm:ss format
                        hours = int(start_time // 3600)
                        minutes = int((start_time % 3600) // 60)
                        seconds = start_time % 60
                        timestamp = f"{hours:02}:{minutes:02}:{seconds:05.2f}"
                        timestamps.append(timestamp)
                        sentences.append(match.group(3).strip())
                    else:
                        print(f"⚠️ Unmatched line format: {line}")  # Debug print for unmatched lines
    except FileNotFoundError:
        print(f"❌ Error: The file '{file_path}' was not found.")
        return [], []

    if not sentences:
        print("⚠️ Warning: No sentences found. Check if the file is empty or if the format is incorrect.")
    return sentences, timestamps

# Extract sentences and timestamps
input_file = 'transcript.txt'
sentences, timestamps = extract_sentences_and_timestamps(input_file)

# Debug: Print extracted sentences and timestamps
print("\nExtracted Sentences:", sentences)
print("Extracted Timestamps:", timestamps)

# Check if sentences are empty
if not sentences:
    print("❌ Error: No valid sentences were extracted. Exiting the script.")
else:
    # Generate sentence embeddings
    embeddings = model.encode(sentences)

    # Debug: Check if embeddings are generated correctly
```

```
    if embeddings.size == 0:
        print("❌ Error: No embeddings were generated. Check if the input text is valid.")
    else:
        # Determine number of clusters based on sentence count
        # Here, we aim for topic-based segmentation, so we adjust clusters based on transcript length
        num_clusters = max(1, len(sentences) // 5)  # Example: 1 cluster per 5 sentences
        kmeans = KMeans(n_clusters=num_clusters, random_state=0)
        labels = kmeans.fit_predict(embeddings)

        # Create segmented output with timestamps
        segments = [[] for _ in range(num_clusters)]
        for i, label in enumerate(labels):
            segments[label].append(f"[{timestamps[i]}] {sentences[i]}")

        # Save segmented output to a new file
        output_file = 'segmented_transcript.txt'
        with open(output_file, 'w') as file:
            for idx, segment in enumerate(segments):
                file.write(f"--- Topic Segment {idx + 1} ---\n")
                file.write("\n".join(segment))
                file.write("\n\n")

        print(f"✅ Topic-segmented transcript saved to {output_file}")
```

```
⚠ Unmatched line format: 96.399 - 102.32000000000001: the requirement of the
⚠ Unmatched line format: 99.119 - 104.2: industry requirement of Manpower for the
⚠ Unmatched line format: 102.32 - 106.19999999999999: industry however there were no
⚠ Unmatched line format: 104.2 - 107.479: corresponding changes which happened
⚠ Unmatched line format: 106.2 - 111.0: education
⚠ Unmatched line format: 107.479 - 113.32: system so therefore I feel that this was
⚠ Unmatched line format: 111.0 - 116.52: the right time when we brought
⚠ Unmatched line format: 113.32 - 121.79899999999999: in the education policy 2020 honorable
⚠ Unmatched line format: 116.52 - 125.19999999999999: prime minister declared it open in July
⚠ Unmatched line format: 121.799 - 129.399: 2020 and we recently celebrated the
⚠ Unmatched line format: 125.2 - 133.4: fourth anniversary of NEP 2020 ncrf has
⚠ Unmatched line format: 129.399 - 136.8: been brought to implement the intent of
⚠ Unmatched line format: 133.4 - 139.36: 2020 2020 is a policy the NP 2020 is a
⚠ Unmatched line format: 136.8 - 141.44: policy and for implementing a policy you
⚠ Unmatched line format: 139.36 - 144.48000000000002: need a
⚠ Unmatched line format: 141.44 - 149.04: framework now why we call it a framework
⚠ Unmatched line format: 144.48 - 151.16: we call it a framework because this is
⚠ Unmatched line format: 149.04 - 154.35999999999999: very flexible
⚠ Unmatched line format: 151.16 - 159.4: this allows you all the
⚠ Unmatched line format: 154.36 - 161.84: Innovation the way you educate your kids
⚠ Unmatched line format: 159.4 - 165.44: you educate your
⚠ Unmatched line format: 161.84 - 168.48: students still it provides you the basic
⚠ Unmatched line format: 165.44 - 171.959: guidelines the the framework the outer
⚠ Unmatched line format: 168.48 - 174.84: layer it provides and that layer is
⚠ Unmatched line format: 171.959 - 177.84: mostly the enabling
⚠ Unmatched line format: 174.84 - 180.76: layer that is such an enabling layer
⚠ Unmatched line format: 177.84 - 183.48: that it has
⚠ Unmatched line format: 180.76 - 185.879: broken the shackles which were there in
⚠ Unmatched line format: 183.48 - 189.48: the education
⚠ Unmatched line format: 185.879 - 191.959: sector right so yes it is a big
⚠ Unmatched line format: 189.48 - 195.23899999999998: Revolution and this is going to change
⚠ Unmatched line format: 191.959 - 197.68: the way we have been educating our kids
⚠ Unmatched line format: 195.239 - 201.08: and this will be Game Changer uh for
⚠ Unmatched line format: 197.68 - 204.56: rest of the world as well so sir I think
⚠ Unmatched line format: 201.08 - 206.36: um uh let's go with this example of
⚠ Unmatched line format: 204.56 - 209.799: let's say I started off living in a
⚠ Unmatched line format: 206.36 - 212.04000000000002: small 2bhk apartment M and I slowly
⚠ Unmatched line format: 209.799 - 214.15900000000002: developed my kitchen to add to my taste
⚠ Unmatched line format: 212.04 - 216.76: buts in a way that it's convenient for
⚠ Unmatched line format: 214.159 - 219.39999999999998: me right and one fine day you came and
⚠ Unmatched line format: 216.76 - 221.879: you changed my kitchen completely I was
⚠ Unmatched line format: 219.4 - 226.0: using a bicycle I moved to a scooter and
⚠ Unmatched line format: 221.879 - 228.2: a car and now you're asking me to fly
⚠ Unmatched line format: 226.0 - 230.879: and you're giving me an aircraft NEP
⚠ Unmatched line format: 228.2 - 232.83999999999997: sounds more like that for me how do I do
⚠ Unmatched line format: 230.879 - 234.92: the transition I fear that I will I will
⚠ Unmatched line format: 232.84 - 236.239: crash if I use a aeroplane without
⚠ Unmatched line format: 234.92 - 238.35999999999999: training I'm talking about all the
⚠ Unmatched line format: 236.239 - 241.12: teachers in the in the country all the
⚠ Unmatched line format: 238.36 - 244.0: schools in the country we have been
⚠ Unmatched line format: 241.12 - 245.59900000000002: driving buses at Max now we should fly
⚠ Unmatched line format: 244.0 - 250.0: how do we do this
⚠ Unmatched line format: 245.599 - 252.23899999999998: okay look at the requirement of the
⚠ Unmatched line format: 250.0 - 253.72: industry requirement of the industry has
⚠ Unmatched line format: 252.239 - 258.4: been moving very
⚠ Unmatched line format: 253.72 - 262.32: fast the technology is emerging every
⚠ Unmatched line format: 258.4 - 265.03999999999996: day and the industry is moving with that
```

```
from sentence_transformers import SentenceTransformer
from sklearn.cluster import KMeans
import re

```

```python
# Load pre-trained model for sentence embeddings
model = SentenceTransformer('all-MiniLM-L6-v2')

# Function to extract sentences and starting timestamps
def extract_sentences_and_timestamps(file_path):
    sentences = []
    timestamps = []
    # Enhanced regex pattern for range-based timestamps
    pattern = re.compile(r"(\d{1,2}(?:\.\d+)?)\s*-\s*(\d{1,2}(?:\.\d+)?):\s*(.*)")

    try:
        with open(file_path, 'r') as file:
            for line in file:
                line = line.strip()
                if line:
                    match = pattern.match(line)
                    if match:
                        start_time = float(match.group(1))
                        # Convert start time to hh:mm:ss format
                        hours = int(start_time // 3600)
                        minutes = int((start_time % 3600) // 60)
                        seconds = start_time % 60
                        timestamp = f"{hours:02}:{minutes:02}:{seconds:05.2f}"
                        timestamps.append(timestamp)
                        sentences.append(match.group(3).strip())
                    else:
                        print(f"⚠️ Unmatched line format: {line}")  # Debug print for unmatched lines
    except FileNotFoundError:
        print(f"❌ Error: The file '{file_path}' was not found.")
        return [], []

    if not sentences:
        print("⚠️ Warning: No sentences found. Check if the file is empty or if the format is incorrect.")
    return sentences, timestamps

# Extract sentences and timestamps
input_file = 'transcript.txt'
sentences, timestamps = extract_sentences_and_timestamps(input_file)

# Debug: Print extracted sentences and timestamps
print("\nExtracted Sentences:", sentences)
print("Extracted Timestamps:", timestamps)

# Check if sentences are empty
if not sentences:
    print("❌ Error: No valid sentences were extracted. Exiting the script.")
else:
    # Generate sentence embeddings
    embeddings = model.encode(sentences)

    # Debug: Check if embeddings are generated correctly
    if embeddings.size == 0:
        print("❌ Error: No embeddings were generated. Check if the input text is valid.")
    else:
        # Determine number of clusters dynamically based on conceptual shifts
        num_clusters = max(1, len(sentences) // 5)  # Example: 1 cluster per 5 sentences
        kmeans = KMeans(n_clusters=num_clusters, random_state=0)
        labels = kmeans.fit_predict(embeddings)

        # Create segmented output with timestamps
        segments = [[] for _ in range(num_clusters)]
        for i, label in enumerate(labels):
            segments[label].append(f"[{timestamps[i]}] {sentences[i]}")

        # Save segmented output to a new file
        output_file = 'segmented_transcript.txt'
        with open(output_file, 'w') as file:
            for idx, segment in enumerate(segments):
                file.write(f"--- Conceptually Relevant Segment {idx + 1} ---\n")
                file.write("\n".join(segment))
                file.write("\n\n")

        print(f"✅ Conceptually relevant segmented transcript saved to {output_file}")
```

```
⚠️ Unmatched line format: 96.399 - 102.32000000000001: the requirement of the
⚠️ Unmatched line format: 99.119 - 104.2: industry requirement of Manpower for the
⚠️ Unmatched line format: 102.32 - 106.19999999999999: industry however there were no
⚠️ Unmatched line format: 104.2 - 107.479: corresponding changes which happened
⚠️ Unmatched line format: 106.2 - 111.0: education
⚠️ Unmatched line format: 107.479 - 113.32: system so therefore I feel that this was
⚠️ Unmatched line format: 111.0 - 116.52: the right time when we brought
⚠️ Unmatched line format: 113.32 - 121.79899999999999: in the education policy 2020 honorable
⚠️ Unmatched line format: 116.52 - 125.19999999999999: prime minister declared it open in July
```

```
⚠️ Unmatched line format: 121.799 - 129.399: 2020 and we recently celebrated the
⚠️ Unmatched line format: 125.2 - 133.4: fourth anniversary of NEP 2020 ncrf has
⚠️ Unmatched line format: 129.399 - 136.8: been brought to implement the intent of
⚠️ Unmatched line format: 133.4 - 139.36: 2020 2020 is a policy the NP 2020 is a
⚠️ Unmatched line format: 136.8 - 141.44: policy and for implementing a policy you
⚠️ Unmatched line format: 139.36 - 144.48000000000002: need a
⚠️ Unmatched line format: 141.44 - 149.04: framework now why we call it a framework
⚠️ Unmatched line format: 144.48 - 151.16: we call it a framework because this is
⚠️ Unmatched line format: 149.04 - 154.35999999999999: very flexible
⚠️ Unmatched line format: 151.16 - 159.4: this allows you all the
⚠️ Unmatched line format: 154.36 - 161.84: Innovation the way you educate your kids
⚠️ Unmatched line format: 159.4 - 165.44: you educate your
⚠️ Unmatched line format: 161.84 - 168.48: students still it provides you the basic
⚠️ Unmatched line format: 165.44 - 171.959: guidelines the the framework the outer
⚠️ Unmatched line format: 168.48 - 174.84: layer it provides and that layer is
⚠️ Unmatched line format: 171.959 - 177.84: mostly the enabling
⚠️ Unmatched line format: 174.84 - 180.76: layer that is such an enabling layer
⚠️ Unmatched line format: 177.84 - 183.48: that it has
⚠️ Unmatched line format: 180.76 - 185.879: broken the shackles which were there in
⚠️ Unmatched line format: 183.48 - 189.48: the education
⚠️ Unmatched line format: 185.879 - 191.959: sector right so yes it is a big
⚠️ Unmatched line format: 189.48 - 195.23899999999998: Revolution and this is going to change
⚠️ Unmatched line format: 191.959 - 197.68: the way we have been educating our kids
⚠️ Unmatched line format: 195.239 - 201.08: and this will be Game Changer uh for
⚠️ Unmatched line format: 197.68 - 204.56: rest of the world as well so sir I think
⚠️ Unmatched line format: 201.08 - 206.36: um uh let's go with this example of
⚠️ Unmatched line format: 204.56 - 209.799: let's say I started off living in a
⚠️ Unmatched line format: 206.36 - 212.04000000000002: small 2bhk apartment M and I slowly
⚠️ Unmatched line format: 209.799 - 214.15900000000002: developed my kitchen to add to my taste
⚠️ Unmatched line format: 212.04 - 216.76: buts in a way that it's convenient for
⚠️ Unmatched line format: 214.159 - 219.39999999999998: me right and one fine day you came and
⚠️ Unmatched line format: 216.76 - 221.879: you changed my kitchen completely I was
⚠️ Unmatched line format: 219.4 - 226.0: using a bicycle I moved to a scooter and
⚠️ Unmatched line format: 221.879 - 228.2: a car and now you're asking me to fly
⚠️ Unmatched line format: 226.0 - 230.879: and you're giving me an aircraft NEP
⚠️ Unmatched line format: 228.2 - 232.83999999999997: sounds more like that for me how do I do
⚠️ Unmatched line format: 230.879 - 234.92: the transition I fear that I will I will
⚠️ Unmatched line format: 232.84 - 236.239: crash if I use a aeroplane without
⚠️ Unmatched line format: 234.92 - 238.35999999999999: training I'm talking about all the
⚠️ Unmatched line format: 236.239 - 241.12: teachers in the in the country all the
⚠️ Unmatched line format: 238.36 - 244.0: schools in the country we have been
⚠️ Unmatched line format: 241.12 - 245.59900000000002: driving buses at Max now we should fly
⚠️ Unmatched line format: 244.0 - 250.0: how do we do this
⚠️ Unmatched line format: 245.599 - 252.23899999999998: okay look at the requirement of the
⚠️ Unmatched line format: 250.0 - 253.72: industry requirement of the industry has
⚠️ Unmatched line format: 252.239 - 258.4: been moving very
⚠️ Unmatched line format: 253.72 - 262.32: fast the technology is emerging every
```

```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
```

```
pip install texttiling_segmenter.py
```

```
ERROR: Could not find a version that satisfies the requirement texttiling_segmenter.py (from versions: none)
ERROR: No matching distribution found for texttiling_segmenter.py
```

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.tokenize.texttiling import TextTilingTokenizer
from google.colab import files  # For file upload

# Ensure you have the necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')

def preprocess_text(text):
    """
    Preprocess the text by tokenizing, lowercasing, and removing stopwords.
    """
    stop_words = set(stopwords.words('english'))
    sentences = sent_tokenize(text)
    words = [word_tokenize(sentence.lower()) for sentence in sentences]
    words = [[word for word in sentence if word.isalnum() and word not in stop_words] for sentence in words]
    return sentences, words

def segment_transcript(file_path):
    """
    Read a transcript from a file, preprocess it, and apply TextTiling to segment it.
```

```
    """
    # Read the file
    with open(file_path, 'r', encoding='utf-8') as file:
        transcript = file.read()

    # Preprocess the text
    sentences, words = preprocess_text(transcript)

    # Apply TextTiling
    tt = TextTilingTokenizer()
    segments = tt.tokenize(transcript)

    # Print the segments
    for i, segment in enumerate(segments):
        print(f"Segment {i+1}:\n{segment}\n")

# Upload the file to Colab
uploaded = files.upload()

# Get the file name
file_name = list(uploaded.keys())[0]

# Segment the transcript
segment_transcript(file_name)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Kalsi sir transcript.txt to Kalsi sir transcript (1).txt
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/nltk/tokenize/texttiling.py in _create_token_table(self, token_sequences, par_breaks)
    242             try:
--> 243                 current_par_break = next(pb_iter)  # skip break at 0
    244             except StopIteration as e:

StopIteration:

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call last)
```

3 frames

```
/usr/local/lib/python3.11/dist-packages/nltk/tokenize/texttiling.py in _create_token_table(self, token_sequences, par_breaks)
    243                 current_par_break = next(pb_iter)  # skip break at 0
    244             except StopIteration as e:
--> 245                 raise ValueError(
    246                     "No paragraph breaks were found(text too short perhaps?)"
    247                 ) from e

ValueError: No paragraph breaks were found(text too short perhaps?)
```

```
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.tokenize.texttiling import TextTilingTokenizer
from google.colab import files

# Download required NLTK resources
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')

def preprocess_text(text):
    """
    Preprocess the text by tokenizing, lowercasing, and removing stopwords.
    """
    stop_words = set(stopwords.words('english'))
```

```
    sentences = sent_tokenize(text)
    words = [word_tokenize(sentence.lower()) for sentence in sentences]
    words = [[word for word in sentence if word.isalnum() and word not in stop_words] for sentence in words]
    return sentences, words

def insert_paragraph_breaks(text, sentences_per_paragraph=5):
    """
    Insert paragraph breaks after every `sentences_per_paragraph` sentences.
    """
    sentences = sent_tokenize(text)
    paragraphs = []
    for i in range(0, len(sentences), sentences_per_paragraph):
        paragraph = " ".join(sentences[i:i + sentences_per_paragraph])
        paragraphs.append(paragraph)
    return "\n\n".join(paragraphs)

def segment_transcript(file_path):
    """
    Read a transcript from a file, preprocess it, and apply TextTiling to segment it.
    """
    # Read the file
    with open(file_path, 'r', encoding='utf-8') as file:
        transcript = file.read()

    # Insert paragraph breaks
    transcript_with_breaks = insert_paragraph_breaks(transcript)

    # Preprocess the text
    sentences, words = preprocess_text(transcript_with_breaks)

    # Apply TextTiling
    tt = TextTilingTokenizer()
    segments = tt.tokenize(transcript_with_breaks)

    # Print the segments
    for i, segment in enumerate(segments):
        print(f"Segment {i+1}:\n{segment}\n")

# Upload the file to Colab
uploaded = files.upload()

# Get the file name
file_name = list(uploaded.keys())[0]

# Segment the transcript
segment_transcript(file_name)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Kalsi sir transcript.txt to Kalsi sir transcript (2).txt
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/nltk/tokenize/texttiling.py in _create_token_table(self, token_sequences, par_breaks)
    242             try:
--> 243                 current_par_break = next(pb_iter)  # skip break at 0
    244             except StopIteration as e:

StopIteration:

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call last)
```

3 frames

```
/usr/local/lib/python3.11/dist-packages/nltk/tokenize/texttiling.py in _create_token_table(self, token_sequences, par_breaks)
    243                 current_par_break = next(pb_iter)  # skip break at 0
    244             except StopIteration as e:
--> 245                 raise ValueError(
    246                     "No paragraph breaks were found(text too short perhaps?)"
    247                 ) from e

ValueError: No paragraph breaks were found(text too short perhaps?)
```

```
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

```
import numpy as np

def preprocess_text(text):
    """
    Preprocess the text by splitting it into sentences and cleaning.
    """
    # Split into sentences
    sentences = re.split(r'(?<=[.!?])\s+', text)
    # Remove short sentences (optional)
    sentences = [s.strip() for s in sentences if len(s.split()) > 5]
    return sentences

def identify_topics(sentences, num_topics=5):
    """
    Identify topics in the text using Latent Dirichlet Allocation (LDA).
    """
    # Create a document-term matrix
    vectorizer = CountVectorizer(stop_words='english', max_df=0.95, min_df=2)
    doc_term_matrix = vectorizer.fit_transform(sentences)

    # Fit LDA model
    lda = LatentDirichletAllocation(n_components=num_topics, random_state=42)
    lda.fit(doc_term_matrix)

    # Get topic distribution for each sentence
    topic_distribution = lda.transform(doc_term_matrix)
    topic_labels = np.argmax(topic_distribution, axis=1)

    # Get top words for each topic
    feature_names = vectorizer.get_feature_names_out()
    topics = []
    for topic_idx, topic in enumerate(lda.components_):
        top_words = [feature_names[i] for i in topic.argsort()[-5:]]
        topics.append(f"Topic {topic_idx + 1}: {', '.join(top_words)}")

    return topic_labels, topics

def segment_transcript(input_file, output_file, num_topics=5):
    """
    Read a transcript from a text file, analyze it, and generate segmented output.
    """
    # Read the input file
    with open(input_file, 'r', encoding='utf-8') as file:
        transcript = file.read()

    # Preprocess the text
    sentences = preprocess_text(transcript)

    # Identify topics
    topic_labels, topics = identify_topics(sentences, num_topics)

    # Group sentences by topic
    segments = {}
    for i, label in enumerate(topic_labels):
        if label not in segments:
            segments[label] = []
        segments[label].append(sentences[i])

    # Write the segmented output to a file
    with open(output_file, 'w', encoding='utf-8') as file:
        for label, segment_sentences in segments.items():
            file.write(f"Segment {label + 1}: {topics[label]}\n")
            file.write("\n".join(segment_sentences) + "\n\n")

# Input and output file paths
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
output_file = "segmented_transcript.txt"  # Output file path

# Segment the transcript
segment_transcript(input_file, output_file, num_topics=5)

print(f"Segmented transcript saved to {output_file}")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-2d212f11d399> in <cell line: 0>()
     71
     72 # Segment the transcript
---> 73 segment_transcript(input_file, output_file, num_topics=5)
     74
     75 print(f"Segmented transcript saved to {output_file}")

                              3 frames
/usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py in fit_transform(self, raw_documents, y)
   1384             min_doc_count = min_df if isinstance(min_df, Integral) else min_df * n_doc
   1385             if max_doc_count < min_doc_count:
-> 1386                 raise ValueError("max_df corresponds to < documents than min_df")
   1387             if max_features is not None:
   1388                 X = self._sort_features(X, vocabulary)

ValueError: max_df corresponds to < documents than min_df
```

```python
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import numpy as np

def preprocess_text(text):
    """
    Preprocess the text by splitting it into sentences and cleaning.
    """
    # Split into sentences
    sentences = re.split(r'(?<=[.!?])\s+', text)
    # Remove short sentences (optional)
    sentences = [s.strip() for s in sentences if len(s.split()) > 5]
    return sentences

def identify_topics(sentences, num_topics=5):
    """
    Identify topics in the text using Latent Dirichlet Allocation (LDA).
    """
    # Create a document-term matrix
    vectorizer = CountVectorizer(stop_words='english', max_df=1.0, min_df=1)
    doc_term_matrix = vectorizer.fit_transform(sentences)

    # Fit LDA model
    lda = LatentDirichletAllocation(n_components=num_topics, random_state=42)
    lda.fit(doc_term_matrix)

    # Get topic distribution for each sentence
    topic_distribution = lda.transform(doc_term_matrix)
    topic_labels = np.argmax(topic_distribution, axis=1)

    # Get top words for each topic
    feature_names = vectorizer.get_feature_names_out()
    topics = []
    for topic_idx, topic in enumerate(lda.components_):
        top_words = [feature_names[i] for i in topic.argsort()[-5:]]
        topics.append(f"Topic {topic_idx + 1}: {', '.join(top_words)}")

    return topic_labels, topics

def segment_transcript(input_file, output_file, num_topics=5):
    """
    Read a transcript from a text file, analyze it, and generate segmented output.
    """
    # Read the input file
    with open(input_file, 'r', encoding='utf-8') as file:
        transcript = file.read()

    # Preprocess the text
    sentences = preprocess_text(transcript)

    # Check if there are enough sentences for topic modeling
    if len(sentences) < num_topics:
        raise ValueError(f"Not enough sentences ({len(sentences)}) for {num_topics} topics.")

    # Identify topics
    topic_labels, topics = identify_topics(sentences, num_topics)

    # Group sentences by topic
    segments = {}
    for i, label in enumerate(topic_labels):
```

```
        if label not in segments:
            segments[label] = []
        segments[label].append(sentences[i])

    # Write the segmented output to a file
    with open(output_file, 'w', encoding='utf-8') as file:
        for label, segment_sentences in segments.items():
            file.write(f"Segment {label + 1}: {topics[label]}\n")
            file.write("\n".join(segment_sentences) + "\n\n")

# Input and output file paths
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
output_file = "segmented_transcript.txt"  # Output file path

# Segment the transcript
segment_transcript(input_file, output_file, num_topics=5)

print(f"Segmented transcript saved to {output_file}")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-39c7be34e700> in <cell line: 0>()
     75
     76 # Segment the transcript
---> 77 segment_transcript(input_file, output_file, num_topics=5)
     78
     79 print(f"Segmented transcript saved to {output_file}")

<ipython-input-14-39c7be34e700> in segment_transcript(input_file, output_file, num_topics)
     52     # Check if there are enough sentences for topic modeling
     53     if len(sentences) < num_topics:
---> 54         raise ValueError(f"Not enough sentences ({len(sentences)}) for {num_topics} topics.")
     55
     56     # Identify topics

ValueError: Not enough sentences (2) for 5 topics.
```

```
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import numpy as np

def preprocess_text(text):
    """
    Preprocess the text by splitting it into sentences and cleaning.
    """
    # Split into sentences
    sentences = re.split(r'(?<=[.!?])\s+', text)
    # Remove short sentences (optional)
    sentences = [s.strip() for s in sentences if len(s.split()) > 5]
    return sentences

def identify_topics(sentences, num_topics=2):
    """
    Identify topics in the text using Latent Dirichlet Allocation (LDA).
    """
    # Create a document-term matrix
    vectorizer = CountVectorizer(stop_words='english', max_df=1.0, min_df=1)
    doc_term_matrix = vectorizer.fit_transform(sentences)

    # Fit LDA model
    lda = LatentDirichletAllocation(n_components=num_topics, random_state=42)
    lda.fit(doc_term_matrix)

    # Get topic distribution for each sentence
    topic_distribution = lda.transform(doc_term_matrix)
    topic_labels = np.argmax(topic_distribution, axis=1)

    # Get top words for each topic
    feature_names = vectorizer.get_feature_names_out()
    topics = []
    for topic_idx, topic in enumerate(lda.components_):
        top_words = [feature_names[i] for i in topic.argsort()[-5:]]
        topics.append(f"Topic {topic_idx + 1}: {', '.join(top_words)}")

    return topic_labels, topics

def segment_transcript(input_file, output_file):
    """
    Read a transcript from a text file, analyze it, and generate segmented output.
    """
    # Read the input file
```

```
    with open(input_file, 'r', encoding='utf-8') as file:
        transcript = file.read()

    # Preprocess the text
    sentences = preprocess_text(transcript)

    # Dynamically adjust the number of topics
    num_topics = min(len(sentences), 5)  # Use up to 5 topics or the number of sentences, whichever is smaller
    if num_topics < 1:
        raise ValueError("Not enough sentences for topic modeling.")

    print(f"Using {num_topics} topics for segmentation.")

    # Identify topics
    topic_labels, topics = identify_topics(sentences, num_topics)

    # Group sentences by topic
    segments = {}
    for i, label in enumerate(topic_labels):
        if label not in segments:
            segments[label] = []
        segments[label].append(sentences[i])

    # Write the segmented output to a file
    with open(output_file, 'w', encoding='utf-8') as file:
        for label, segment_sentences in segments.items():
            file.write(f"Segment {label + 1}: {topics[label]}\n")
            file.write("\n".join(segment_sentences) + "\n\n")

# Input and output file paths
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
output_file = "segmented_transcript.txt"  # Output file path

# Segment the transcript
segment_transcript(input_file, output_file)

print(f"Segmented transcript saved to {output_file}")
```

```
Using 2 topics for segmentation.
Segmented transcript saved to segmented_transcript.txt
```

```
import re
from nltk.tokenize.texttiling import TextTilingTokenizer
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize
from nltk import download

# Download necessary NLTK data
download('punkt')
download('stopwords')

def preprocess_text(text):
    """
    Preprocess the text by removing stopwords and tokenizing into sentences.
    """
    stop_words = set(stopwords.words('english'))
    sentences = sent_tokenize(text)
    cleaned_sentences = []
    for sentence in sentences:
        words = [word.lower() for word in re.findall(r'\b\w+\b', sentence) if word.lower() not in stop_words]
        cleaned_sentences.append(" ".join(words))
    return cleaned_sentences

def segment_transcript(input_file, output_file):
    """
    Segment the transcript using the TextTiling algorithm.
    """
    # Read the input file
    with open(input_file, 'r', encoding='utf-8') as file:
        transcript = file.read()

    # Preprocess the text
    sentences = preprocess_text(transcript)

    # Combine sentences into a single string for TextTiling
    text = "\n\n".join(sentences)

    # Initialize TextTilingTokenizer
    tt = TextTilingTokenizer()

    # Segment the text
    try:
```

```
        segments = tt.tokenize(text)
    except ValueError as e:
        print(f"TextTiling failed: {e}")
        print("Falling back to sentence-based segmentation.")
        segments = ["\n".join(sentences)]

    # Write the segmented output to a file
    with open(output_file, 'w', encoding='utf-8') as file:
        for i, segment in enumerate(segments):
            file.write(f"Segment {i + 1}:\n{segment}\n\n")

# Input and output file paths
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
output_file = "segmented_transcript.txt"  # Output file path

# Segment the transcript
segment_transcript(input_file, output_file)

print(f"Segmented transcript saved to {output_file}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Segmented transcript saved to segmented_transcript.txt
```

```
import re
from nltk.tokenize.texttiling import TextTilingTokenizer
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk import download

# Download necessary NLTK data
download('punkt')
download('stopwords')

def preprocess_text(text):
    """
    Preprocess the text by removing stopwords and tokenizing into sentences.
    """
    stop_words = set(stopwords.words('english'))
    sentences = sent_tokenize(text)
    cleaned_sentences = []
    for sentence in sentences:
        words = [word.lower() for word in word_tokenize(sentence) if word.isalnum() and word.lower() not in stop_words]
        cleaned_sentences.append(" ".join(words))
    return cleaned_sentences

def extract_topic(segment, top_n=3):
    """
    Extract the top keywords from a segment to use as the topic label.
    """
    vectorizer = TfidfVectorizer(max_features=10)
    tfidf_matrix = vectorizer.fit_transform([segment])
    feature_names = vectorizer.get_feature_names_out()
    tfidf_scores = tfidf_matrix.toarray()[0]
    top_keywords = [feature_names[i] for i in tfidf_scores.argsort()[-top_n:][::-1]]
    return ", ".join(top_keywords)

def segment_transcript(input_file, output_file):
    """
    Segment the transcript using the TextTiling algorithm and assign topic labels.
    """
    # Read the input file
    with open(input_file, 'r', encoding='utf-8') as file:
        transcript = file.read()

    # Preprocess the text
    sentences = preprocess_text(transcript)

    # Combine sentences into a single string for TextTiling
    text = "\n\n".join(sentences)

    # Initialize TextTilingTokenizer
    tt = TextTilingTokenizer()

    # Segment the text
    try:
        segments = tt.tokenize(text)
    except ValueError as e:
        print(f"TextTiling failed: {e}")
```

```
        print("Falling back to sentence-based segmentation.")
        segments = ["\n".join(sentences)]

    # Write the segmented output to a file with topic labels
    with open(output_file, 'w', encoding='utf-8') as file:
        for i, segment in enumerate(segments):
            # Extract the topic for the segment
            topic = extract_topic(segment)
            file.write(f"Segment {i + 1} - Topic: {topic}\n")
            file.write(f"{segment}\n\n")

# Input and output file paths
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
output_file = "segmented_transcript.txt"  # Output file path

# Segment the transcript
segment_transcript(input_file, output_file)

print(f"Segmented transcript saved to {output_file}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Segmented transcript saved to segmented_transcript.txt
```

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string

nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

def preprocess(text):
    # Tokenize
    tokens = word_tokenize(text.lower())

    # Remove punctuation and stopwords
    tokens = [word for word in tokens if word.isalnum() and word not in stopwords.words('english')]

    # Lemmatize
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return tokens

# Example transcript
transcript = """
Your transcript text goes here. It can be multiple paragraphs long.
This is just an example to show how the preprocessing works.
"""

# Preprocess the transcript
processed_transcript = preprocess(transcript)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
# Create a dictionary and corpus
dictionary = Dictionary([processed_transcript])
corpus = [dictionary.doc2bow(processed_transcript)]

# Train an LDA model
lda_model = LdaModel(corpus, num_topics=5, id2word=dictionary, passes=15)

# Tokenize the transcript into sentences
sentences = sent_tokenize(transcript)

# Segment the transcript based on topics
```

```
segments = []
for sentence in sentences:
    bow = dictionary.doc2bow(preprocess(sentence))
    topic_distribution = lda_model.get_document_topics(bow)
    dominant_topic = max(topic_distribution, key=lambda x: x[1])[0]
    segments.append((sentence, dominant_topic))

# Write the segments to a text file
with open("segmented_transcript_lda.txt", "w") as output_file:
    for sentence, topic in segments:
        output_file.write(f"Topic {topic}: {sentence}\n\n")

print("Segmentation complete. Output saved to 'segmented_transcript_lda.txt'.")
```

```
from gensim.corpora import Dictionary  # Import Dictionary from gensim.corpora
```

```
from gensim.models import LdaModel
```

```
from nltk.tokenize import word_tokenize, sent_tokenize
```

```
# Preprocessing function
def preprocess(text):
    tokens = word_tokenize(text.lower())
    tokens = [word for word in tokens if word.isalnum() and word not in stopwords.words('english')]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return tokens

# Load the transcript from a file
with open("Kalsi sir transcript.txt", "r") as file:
    transcript = file.read()

# Preprocess the transcript
processed_transcript = preprocess(transcript)

# Create a dictionary and corpus
dictionary = Dictionary([processed_transcript])
corpus = [dictionary.doc2bow(processed_transcript)]

# Train an LDA model
lda_model = LdaModel(corpus, num_topics=5, id2word=dictionary, passes=15)

# Tokenize the transcript into sentences
sentences = sent_tokenize(transcript)

# Segment the transcript based on topics
segments = []
for sentence in sentences:
    bow = dictionary.doc2bow(preprocess(sentence))
    topic_distribution = lda_model.get_document_topics(bow)
    dominant_topic = max(topic_distribution, key=lambda x: x[1])[0]
    segments.append((sentence, dominant_topic))

# Write the segments to a text file
with open("segmented_transcript_lda.txt", "w") as output_file:
    for sentence, topic in segments:
        output_file.write(f"Topic {topic}: {sentence}\n\n")

print("Segmentation complete. Output saved to 'segmented_transcript_lda.txt'.")
```

```
Segmentation complete. Output saved to 'segmented_transcript_lda.txt'.
```

```
pip install bertopic
```

```
Collecting bertopic
  Downloading bertopic-0.16.4-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: hdbscan>=0.8.29 in /usr/local/lib/python3.11/dist-packages (from bertopic) (0.8.40)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.11/dist-packages (from bertopic) (1.26.4)
Requirement already satisfied: pandas>=1.1.5 in /usr/local/lib/python3.11/dist-packages (from bertopic) (2.2.2)
Requirement already satisfied: plotly>=4.7.0 in /usr/local/lib/python3.11/dist-packages (from bertopic) (5.24.1)
Requirement already satisfied: scikit-learn>=0.22.2.post1 in /usr/local/lib/python3.11/dist-packages (from bertopic) (1.6.1)
Requirement already satisfied: sentence-transformers>=0.4.1 in /usr/local/lib/python3.11/dist-packages (from bertopic) (3.4.1)
Requirement already satisfied: tqdm>=4.41.1 in /usr/local/lib/python3.11/dist-packages (from bertopic) (4.67.1)
Requirement already satisfied: umap-learn>=0.5.0 in /usr/local/lib/python3.11/dist-packages (from bertopic) (0.5.7)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.11/dist-packages (from hdbscan>=0.8.29->bertopic) (1.13.1)
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.11/dist-packages (from hdbscan>=0.8.29->bertopic) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.5->bertopic)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.5->bertopic) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.5->bertopic) (2025.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly>=4.7.0->bertopic) (9.0.0)
```

```
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly>=4.7.0->bertopic) (24.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.22.2.post1->
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers>=0.4.1->berto
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers>=0
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers>=0.4.1->bertopic) (1
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.11/dist-packages (from umap-learn>=0.5.0->bertopic) (0.60
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.11/dist-packages (from umap-learn>=0.5.0->bertopic) (0
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transf
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentenc
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-tra
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transf
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=0.51.2->umap-le
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.1.5->b
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers>=0
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers>=0.4
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentenc
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers>=0.4.1->bertopic)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformer
```

```python
from bertopic import BERTopic

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
with open(input_file, "r") as file:
    transcript = file.read()

# Split the transcript into sentences
sentences = transcript.split('. ')  # Simple split for demonstration
sentences = [s.strip() for s in sentences if s.strip()]

# Train BERTopic model
topic_model = BERTopic()
topics, _ = topic_model.fit_transform(sentences)

# Get topic information
topic_info = topic_model.get_topic_info()

# Save the topics to a file
output_file = "bertopic_clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for topic_id in topic_info['Topic']:
        if topic_id != -1:  # Ignore outliers
            f.write(f"Topic {topic_id}:\n")
            for sentence, assigned_topic in zip(sentences, topics):
                if assigned_topic == topic_id:
                    f.write(f" - {sentence}\n")
            f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```

```
modules.json: 100%                                          349/349 [00:00<00:00, 22.5kB/s]
config_sentence_transformers.json: 100%                     116/116 [00:00<00:00, 9.46kB/s]
README.md: 10.5k/10.5k [00:00<00:00, 864kB/s]
sentence_bert_config.json: 100%                             53.0/53.0 [00:00<00:00, 4.68kB/s]
config.json: 612/612 [00:00<00:00, 36.2kB/s]
model.safetensors: 100%                                     90.9M/90.9M [00:01<00:00, 105MB/s]
tokenizer_config.json: 100%                                 350/350 [00:00<00:00, 22.0kB/s]
vocab.txt: 232k/232k [00:00<00:00, 4.22MB/s]
tokenizer.json: 100%                                        466k/466k [00:00<00:00, 17.6MB/s]
special_tokens_map.json: 100%                               112/112 [00:00<00:00, 6.22kB/s]
config.json: 190/190 [00:00<00:00, 8.95kB/s]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-12c776d8ef88> in <cell line: 0>()
     12 # Train BERTopic model
     13 topic_model = BERTopic()
---> 14 topics, _ = topic_model.fit_transform(sentences)
     15
     16 # Get topic information

                              5 frames
/usr/local/lib/python3.11/dist-packages/numpy/core/_methods.py in _amax(a, axis, out, keepdims, initial, where)
     39 def _amax(a, axis=None, out=None, keepdims=False,
     40           initial=_NoValue, where=True):
---> 41     return umr_maximum(a, axis, None, out, keepdims, initial, where)
     42
     43 def _amin(a, axis=None, out=None, keepdims=False,

ValueError: zero-size array to reduction operation maximum which has no identity
```

```
from bertopic import BERTopic
import nltk

# Download the Punkt tokenizer for sentence splitting
nltk.download('punkt')

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
with open(input_file, "r") as file:
    transcript = file.read()

# Split the transcript into sentences using NLTK's sent_tokenize
sentences = nltk.sent_tokenize(transcript)  # More robust sentence splitting
sentences = [s.strip() for s in sentences if s.strip()]  # Remove empty strings

# Debugging: Print the sentences to verify
print("Sentences extracted from the transcript:")
for i, sentence in enumerate(sentences):
    print(f"{i + 1}: {sentence}")

# Check if sentences are empty
if not sentences:
    raise ValueError("No sentences found in the transcript. Please check the input file.")

# Train BERTopic model
topic_model = BERTopic()
topics, _ = topic_model.fit_transform(sentences)

# Get topic information
topic_info = topic_model.get_topic_info()

# Save the topics to a file
output_file = "bertopic_clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for topic_id in topic_info['Topic']:
        if topic_id != -1:  # Ignore outliers
            f.write(f"Topic {topic_id}:\n")
            for sentence, assigned_topic in zip(sentences, topics):
                if assigned_topic == topic_id:
                    f.write(f" - {sentence}\n")
            f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Sentences extracted from the transcript:
1: so sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the
I think immediately after you play the next big revolution personally I think is an education and the complete homework for this
do you think ncrf plus any people together will be the next big revolution after up in India absolutely and why I think so is bec
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
July 2020 and we recently celebrated the 4th anniversary of NP 2020 ncrf has been brought to implement the intent of 2020 2020 th
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guide.
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
pull up my kitchen to add to my taste buds in a way that it's convenient for me and one fine day you came and you changed my kitc
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will you be not only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we ar
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyin
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for nc
it allows for creditor of all learnings weather in academics orange killing or an experiential learning and all these three kinds
and people who are already skilled or already in the professional area not there the experiential learning would play a big part
increase of technology which has been created by single Department good question so therefore all of us we have to work in tander
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills y
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of all
have you already seen the Fallout of this know you can see the photo you can see how many Engineers are there for Designing our b
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework ho
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept pract
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those ar
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 yea
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so th
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test t
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumin
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that w
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessme
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assessm
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and t
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you fe
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-minded
[Unintelligible]
play nursery and she says it lightly and the answer lies in this new framework and know how many times as you rightly say that st
it can be adopted but colleges that are approved by a city let's say a state Technical University may not know these rules of cou
of course a price them go and talk to them and then educate them but over and Beyond this there are some subtle problems that I n
play by 50% or I'll engage my faculty for the betterment of my students at a level which need not necessarily be teaching in the
picture Affiliated to these 1200 University leaving all the ionis that is Institute of national importance but they have about 4
2: 3 4 4
do you think this ratio is sufficient to teach a technical subject know once we are sending our students out again these teachers
it's not easy to create a project which is outcome based creating that project itself is going to take a lot of time similarly th
[Unintelligible]
challenge dead and then they're going to be no nutrition I suppose it is really against the students should not disincentivize re
what clothes should not be in the classroom but outside the classroom evaluation and teacher would be much happier teacher would
```

```
from bertopic import BERTopic
import nltk

# Download the Punkt tokenizer for sentence splitting
nltk.download('punkt')

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
with open(input_file, "r") as file:
    transcript = file.read()

# Check if the transcript is empty
if not transcript.strip():
    raise ValueError("The input file is empty. Please provide a valid transcript.")

# Split the transcript into sentences using NLTK's sent_tokenize
sentences = nltk.sent_tokenize(transcript)  # More robust sentence splitting
sentences = [s.strip() for s in sentences if s.strip()]  # Remove empty strings

# Debugging: Print the sentences to verify
print("Sentences extracted from the transcript:")
for i, sentence in enumerate(sentences):
    print(f"{i + 1}: {sentence}")

# Check if sentences are empty
if not sentences:
    raise ValueError("No sentences found in the transcript. Please check the input file.")

# Train BERTopic model
```

```
topic_model = BERTopic()
topics, _ = topic_model.fit_transform(sentences)

# Get topic information
topic_info = topic_model.get_topic_info()

# Save the topics to a file
output_file = "bertopic_clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for topic_id in topic_info['Topic']:
        if topic_id != -1:  # Ignore outliers
            f.write(f"Topic {topic_id}:\n")
            for sentence, assigned_topic in zip(sentences, topics):
                if assigned_topic == topic_id:
                    f.write(f" - {sentence}\n")
            f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
Sentences extracted from the transcript:
1: sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the
Think immediately after in my play the next big revolution personally I think is in education and the complete homework for this
do you think ncrf plus any people together will be the next big revolution after us in India absolutely and why I think so is be
and we have to integrate these skills in the courses these skills cannot be just taught on a standalone basis yes you can explai
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
to talk and interact with students why are we really bothered about that cognitive time the teacher is one who is designed his
July 2020 and yet we recently celebrated 8th anniversary of NEP 2020 and it has been bought along with the students of 2020 2021 th
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guide
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
pulp in my kitchen 456 add coffee in a tea time in a way that it's convenient for me and one fine day you came and you changed my kit
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will go beyond only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we a
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyi
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for n
and people who are already skilled or already in the professional area not there the experiential learning and all these three kind
increase of technology which has been created by single Department good question so therefore all of us we have to work in tande
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of al
have you already seen the fallout of this know you can see the photo you can see how many Engineers are there for Designing our
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework h
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept prac
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those a
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 ye
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so t
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumi
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessm
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assess
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you f
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-mind
[Unintelligible]
play nursery and she says it lightly and the answer lies in this new framework and know how many times as you rightly say that s
it can be adopted but colleges that are approved by a city let's say a state Technical University may not know these rules of co
of course a price them go and talk to them and then educate them but over and Beyond this there are some subtle problems that I
play by 50% or I'll engage my faculty for the betterment of my students at a level which need not necessarily be teaching in the
picture Affiliated to these 1200 University leaving all the ionis that is Institute of national importance but they have about 4
2: 3 4 4
do you think this ratio is sufficient to teach a technical subject know once we are sending our students out again these teacher
it's not easy to create a project which is outcome based creating that project itself is going to take a lot of time similarly t
[Unintelligible]
challenge dead and then they're going to be no nutrition I suppose it is really against the students should not disincentivize r
what clothes should not be in the classroom but outside the classroom evaluation and teacher would be much happier teacher would
```

```
#BERTOPIC
```

```
from bertopic import BERTopic
import nltk

# Download the Punkt tokenizer for sentence splitting
nltk.download('punkt')

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
```

```
# Train BERTopic model
topic_model = BERTopic()
topics, _ = topic_model.fit_transform(sentences)

# Get topic information
topic_info = topic_model.get_topic_info()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/numpy/core/_methods.py in _amax(a, axis, out, keepdims, initial, where)
     39 def _amax(a, axis=None, out=None, keepdims=False,
     40          initial=_NoValue, where=True):
---> 41     return umr_maximum(a, axis, None, out, keepdims, initial, where)
     42
     43 def _amin(a, axis=None, out=None, keepdims=False,
ValueError: zero-size array to reduction operation maximum which has no identity
```

```
with open(input_file, "r") as file:
    transcript = file.read()

# Check if the transcript is empty
if not transcript.strip():
    raise ValueError("The input file is empty. Please provide a valid transcript.")

# Split the transcript into sentences using NLTK's sent_tokenize
sentences = nltk.sent_tokenize(transcript)  # More robust sentence splitting
sentences = [s.strip() for s in sentences if s.strip()]  # Remove empty strings

# Debugging: Print the sentences to verify
print("Sentences extracted from the transcript:")
for i, sentence in enumerate(sentences):
    print(f"{i + 1}: {sentence}")

# Check if sentences are empty
if not sentences:
    raise ValueError("No sentences found in the transcript. Please check the input file.")

# Train BERTopic model
topic_model = BERTopic()
topics, _ = topic_model.fit_transform(sentences)

# Get topic information
topic_info = topic_model.get_topic_info()

# Save the topics to a file
output_file = "bertopic_clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for topic_id in topic_info['Topic']:
        if topic_id != -1:  # Ignore outliers
            f.write(f"Topic {topic_id}:\n")
            for sentence, assigned_topic in zip(sentences, topics):
                if assigned_topic == topic_id:
                    f.write(f" - {sentence}\n")
            f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```

```
so we picked up the critical learning outcomes the generic learning outcomes which every student must gain as a graduate so The (
play level descriptors based on a particular soft skills you will get a particular level in the NCR that course will be related e
there are six kinds of thinking itself there is adaptive thinking computational thinking critical thinking then you have creative
and we have to integrate those skills in the courses these skills cannot be just caught on a standalone basis yes you can explain
to talk and interact with students why are we really bothered about that cognitive time the teacher is one who is dedicated his 
it'll be a very thoughtful process cognitive process through which he will also be enjoying along with the students he will feel
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-ed92283479ab> in <cell line: 0>()
     29 # Train BERTopic model
     30 topic_model = BERTopic()
---> 31 topics, _ = topic_model.fit_transform(sentences)
     32
     33 # Get topic information

                              5 frames
/usr/local/lib/python3.11/dist-packages/numpy/core/_methods.py in _amax(a, axis, out, keepdims, initial, where)
     39 def _amax(a, axis=None, out=None, keepdims=False,
     40           initial=_NoValue, where=True):
---> 41     return umr_maximum(a, axis, None, out, keepdims, initial, where)
     42
     43 def _amin(a, axis=None, out=None, keepdims=False,

ValueError: zero-size array to reduction operation maximum which has no identity
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Sentences extracted from the transcript:
1: so sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the
I think immediately after you play the next big revolution personally I think is an education and the complete homework for this
do you think ncrf plus any people together will be the next big revolution after up in India absolutely and why I think so is bec
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
July 2020 and we recently celebrated the 4th anniversary of NP 2020 ncrf has been brought to implement the intent of 2020 2020 th
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guide.
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
pull up my kitchen to add to my taste buds in a way that it's convenient for me and one fine day you came and you changed my kitc
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will you be not only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we ar
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyin
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for nc
it allows for creditor of all learnings weather in academics orange killing or an experiential learning and all these three kinds
and people who are already skilled or already in the professional area not there the experiential learning would play a big part
increase of technology which has been created by single Department good question so therefore all of us we have to work in tander
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills y
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of all
have you already seen the Fallout of this know you can see the photo you can see how many Engineers are there for Designing our b
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework ho
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept pract
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those ar
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 yea
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so th
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test t
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumin
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that w
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessme
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assessm
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and t
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you fe
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-minded
[Unintelligible]
play nursery and she says it lightly and the answer lies in this new framework and know how many times as you rightly say that st
it can be adopted but colleges that are approved by a city let's say a state Technical University may not know these rules of cou
of course a price them go and talk to them and then educate them but over and Beyond this there are some subtle problems that I n
play by 50% or I'll engage my faculty for the betterment of my students at a level which need not necessarily be teaching in the
picture Affiliated to these 1200 University leaving all the ionis that is Institute of national importance but they have about 4
2: 3 4 4
do you think this ratio is sufficient to teach a technical subject know once we are sending our students out again these teachers
it's not easy to create a project which is outcome based creating that project itself is going to take a lot of time similarly th
[Unintelligible]
challenge dead and then they're going to be no nutrition I suppose it is really against the students should not disincentivize re
what clothes should not be in the classroom but outside the classroom evaluation and teacher would be much happier teacher would
```

```
from bertopic import BERTopic
import nltk

# Download the Punkt tokenizer for sentence splitting
nltk.download('punkt')

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
with open(input_file, "r") as file:
    transcript = file.read()

# Check if the transcript is empty
if not transcript.strip():
    raise ValueError("The input file is empty. Please provide a valid transcript.")

# Split the transcript into sentences using NLTK's sent_tokenize
sentences = nltk.sent_tokenize(transcript)  # More robust sentence splitting
sentences = [s.strip() for s in sentences if s.strip()]  # Remove empty strings

# Debugging: Print the sentences to verify
print("Sentences extracted from the transcript:")
for i, sentence in enumerate(sentences):
    print(f"{i + 1}: {sentence}")

# Check if sentences are empty
if not sentences:
    raise ValueError("No sentences found in the transcript. Please check the input file.")

# Train BERTopic model
```

```
topic_model = BERTopic()
topics, _ = topic_model.fit_transform(sentences)

# Get topic information
topic_info = topic_model.get_topic_info()

# Save the topics to a file
output_file = "bertopic_clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for topic_id in topic_info['Topic']:
        if topic_id != -1:  # Ignore outliers
            f.write(f"Topic {topic_id}:\n")
            for sentence, assigned_topic in zip(sentences, topics):
                if assigned_topic == topic_id:
                    f.write(f" - {sentence}\n")
            f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Sentences extracted from the transcript:
1: sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the
do you think ncrf plus any people together will be the next big revolution after us in India absolutely and why I think so is be
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guide
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will you agree on only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we ar
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyin
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for no
it allows for creditor of all learnings weather in academics orange killing or an experiential learning and all these three kinds
and people who are already skilled or already in the professional area not there the experiential learning would play a big part
increase of technology which has been created by single Department good question so therefore all of us we have to work in tander
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills y
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of al
have you already seen the fallout of this know you can see the photo you can see how many Engineers are there for Designing our b
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework ho
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept pract
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those ar
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 yea
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so th
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test t
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumin
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that w
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessme
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assessm
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and t
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you fe
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-minded
[Unintelligible]
play nursery and she says it lightly and the answer lies in this new framework and know how many times as you rightly say that st
it can be adopted but colleges that are approved by a city let's say a state Technical University may not know these rules of cou
of course a price them go and talk to them and then educate them but over and Beyond this there are some subtle problems that I n
play by 50% or I'll engage my faculty for the betterment of my students at a level which need not necessarily be teaching in the
picture Affiliated to these 1200 University leaving all the ionis that is Institute of national importance but they have about 4
2: 3 4 4
do you think this ratio is sufficient to teach a technical subject know once we are sending our students out again these teachers
it's not easy to create a project which is outcome based creating that project itself is going to take a lot of time similarly th
[Unintelligible]
challenge dead and then they're going to be no nutrition I suppose it is really against the students should not disincentivize re
what clothes should not be in the classroom but outside the classroom evaluation and teacher would be much happier teacher would
```

```
pip install openai scikit-learn nltk
```

```
Requirement already satisfied: openai in /usr/local/lib/python3.11/dist-packages (1.61.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: anyio<4,>=3.5.0 in /usr/local/lib/python3.11/dist-packages (3.7.1)
Requirement already satisfied: distro<1,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (1.9.0)
Requirement already satisfied: jiter<0.9,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (0.8.2)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.11/dist-packages (2.10.6)
Requirement already satisfied: sniffio<1 in /usr/local/lib/python3.11/dist-packages (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.11/dist-packages (4.67.1)
Requirement already satisfied: typing-extensions<4,>=4.7 in /usr/local/lib/python3.11/dist-packages (4.12.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (3.6.0)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (8.1.8)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (2024.11.6)
```

```
Requirement already satisfied: tqdm>=4.7.1 in /usr/local/lib/python3.11/dist-packages (from openai) (4.67.1)
Requirement already satisfied: typing-extensions<5,>=4.11 in /usr/local/lib/python3.11/dist-packages (from openai) (4.12.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5,>=3.5.0->openai) (3.10)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->openai) (2025.1.31)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->openai) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->ope
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->openai)
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->openai) (2
```

the attention span of a student nitching actually requires sounds cool but it requires a lotof willpower from the parts store
country are capable of doing this don't you think it's a herculean task that we are proposing I got the spirit of the question r:
who is below is very nice and and the law has just called it was just being discussed in the media this week only.
the rules about short videos short parts in new houses it very learned and they have banned shots for young kids because if they
how to start reading a book and finish a book how to make a pretty out of the book how to share your learning from the book with
not only that India is grappling with this problem and people are trying to find their own solution that is where I think that is
for the shots for three floors I see a bad review in the first part good reds in the second no Babel trying to design if we can giv
he would give it in pieces of 119each I see that as as a wonderful thought but I think the students wants to see the short video
that is where the problem lies that is algorithm is wicked there yes we fix that with a nice algorithm and use fast food to deliv
On the attention span of the employees and this is our conclusion that any employee will not be able to spend more than 7 1/2 mi
that is now it's about 7 1/2 8 minutes total and then you can give him in one setting you can at most give him three concepts so
studied further and write so in that connection there are people who can do it first who are slow Bloomers in the area where we
testing 5 about slow productivity on the other m confused here when we when we when we end so there are some gifted minds always
how many yards in marathon physics night okay and then we also talked about Heatherton student is so smart that he's able to solv (2
most feasible solution Which is closest to the real solution to your character that's the most interesting part of being created
high quality organic knowledge absolutely absolutely if he's able to achieve those learning outcomes whatever happened to find fo
```

```python
import openai
from sklearn.cluster import KMeans
import numpy as np

# Set your OpenAI API key
openai.api_key = "sk-proj-hTAcJedpA-L6e5dEsETts2Smkea8aobk4gRC5FMpUY82gFFquqZItpbxjGBIVPata9A2upge_rT3BlbkFJfAYUkHc6lRQEERO_a7Yw-ai(

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
with open(input_file, "r") as file:
    transcript = file.read()

# Check if the transcript is empty
if not transcript.strip():
    raise ValueError("The input file is empty. Please provide a valid transcript.")

# Split the transcript into sentences using NLTK's sent_tokenize
import nltk
nltk.download('punkt')
sentences = nltk.sent_tokenize(transcript)  # More robust sentence splitting
sentences = [s.strip() for s in sentences if s.strip()]  # Remove empty strings

# Debugging: Print the sentences to verify
print("Sentences extracted from the transcript:")
for i, sentence in enumerate(sentences):
    print(f"{i + 1}: {sentence}")

# Check if sentences are empty
if not sentences:
    raise ValueError("No sentences found in the transcript. Please check the input file.")

# Generate embeddings using OpenAI's text-embedding-ada-002 model
def get_embeddings(texts):
    response = openai.Embedding.create(
        input=texts,
        model="text-embedding-ada-002"
    )
    return [item['embedding'] for item in response['data']]

# Get embeddings for all sentences
sentence_embeddings = get_embeddings(sentences)

# Apply K-Means clustering
num_clusters = 5  # Number of topics
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(sentence_embeddings)
cluster_labels = kmeans.labels_

# Group sentences by cluster
clustered_sentences = {i: [] for i in range(num_clusters)}
for sentence, label in zip(sentences, cluster_labels):
    clustered_sentences[label].append(sentence)

# Save the clusters to a file
output_file = "openai_clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for cluster_id, sentences_in_cluster in clustered_sentences.items():
        f.write(f"Cluster {cluster_id}:\n")
        for sentence in sentences_in_cluster:
            f.write(f" - {sentence}\n")
        f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Sentences extracted from the transcript:
1: so sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the
I think immediately after you play the next big revolution personally I think is an education and the complete homework for this
do you think ncrf plus any people together will be the next big revolution after up in India absolutely and why I think so is bec
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
July 2020 and we recently celebrated the 4th anniversary of NP 2020 ncrf has been brought to implement the intent of 2020 2020 th
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guide.
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
pull up my kitchen to add to my taste buds in a way that it's convenient for me and one fine day you came and you changed my kitc
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will you be not only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we ar
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyin
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for no
it allows for creditor of all learnings weather in academics orange killing or an experiential learning and all these three kinds
and people who are already skilled or already in the professional area not there the experiential learning would play a big part
increase of technology which has been created by single Department good question so therefore all of us we have to work in tander
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills y
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of all
have you already seen the Fallout of this know you can see the photo you can see how many Engineers are there for Designing our b
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework ho
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept pract
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those ar
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 yea
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so th
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test t
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumin
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that w
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessme
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assessm
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and t
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you fe
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-minded
[Unintelligible]
play nursery and she says it lightly and the answer lies in this new framework and know how many times as you rightly say that st
it can be adopted but colleges that are approved by a city let's say a state Technical University may not know these rules of cou
of course a price them go and talk to them and then educate them but over and Beyond this there are some subtle problems that I n
play by 50% or I'll engage my faculty for the betterment of my students at a level which need not necessarily be teaching in the
picture Affiliated to these 1200 University leaving all the ionis that is Institute of national importance but they have about 4
2: 3 4 4
do you think this ratio is sufficient to teach a technical subject know once we are sending our students out again these teachers
it's not easy to create a project which is outcome based creating that project itself is going to take a lot of time similarly th
[Unintelligible]
challenge dead and then they're going to be no nutrition I suppose it is really against the students should not disincentivize re
what clothes should not be in the classroom but outside the classroom evaluation and teacher would be much happier teacher would
```

```
from openai import OpenAI
from sklearn.cluster import KMeans
import numpy as np
import nltk

# Initialize the OpenAI client
client = OpenAI(api_key="sk-proj-hTAcJedpA-L6e5dEsETts2Smkea8aobk4gRC5FMpUY82gFFquqZItpbxjGBIVPata9A2upge_rT3BlbkFJfAYUkHc6lRQEERO_a

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
with open(input_file, "r") as file:
    transcript = file.read()

# Check if the transcript is empty
if not transcript.strip():
    raise ValueError("The input file is empty. Please provide a valid transcript.")

# Split the transcript into sentences using NLTK's sent_tokenize
nltk.download('punkt')
sentences = nltk.sent_tokenize(transcript)  # More robust sentence splitting
sentences = [s.strip() for s in sentences if s.strip()]  # Remove empty strings

# Debugging: Print the sentences to verify
print("Sentences extracted from the transcript:")
for i, sentence in enumerate(sentences):
    print(f"{i + 1}: {sentence}")

# Check if sentences are empty
if not sentences:
```

```
    raise ValueError("No sentences found in the transcript. Please check the input file.")

# Generate embeddings using OpenAI's text-embedding-ada-002 model
def get_embeddings(texts):
    response = client.embeddings.create(
        input=texts,
        model="text-embedding-ada-002"
    )
    return [item.embedding for item in response.data]

# Get embeddings for all sentences
sentence_embeddings = get_embeddings(sentences)

# Apply K-Means clustering
num_clusters = 5  # Number of topics
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(sentence_embeddings)
cluster_labels = kmeans.labels_

# Group sentences by cluster
clustered_sentences = {i: [] for i in range(num_clusters)}
for sentence, label in zip(sentences, cluster_labels):
    clustered_sentences[label].append(sentence)

# Save the clusters to a file
output_file = "openai_clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for cluster_id, sentences_in_cluster in clustered_sentences.items():
        f.write(f"Cluster {cluster_id}:\n")
        for sentence in sentences_in_cluster:
            f.write(f" - {sentence}\n")
        f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```

1 frames

```
/usr/local/lib/python3.11/dist-packages/openai/lib/_old_api.py in __call__(self, *_args, **_kwargs)
     37
     38     def __call__(self, *_args: Any, **_kwargs: Any) -> Any:
---> 39         raise APIRemovedInV1(symbol=self._symbol)
     40
     41

APIRemovedInV1:

You tried to access openai.Embedding, but this is no longer supported in openai>=1.0.0 - see the README at
https://github.com/openai/openai-python for the API.

You can run `openai migrate` to automatically upgrade your codebase to use the 1.0.0 interface.

Alternatively, you can pin your installation to the old version, e.g. `pip install openai==0.28`

A detailed migration guide is available here: https://github.com/openai/openai-python/discussions/742
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Sentences extracted from the transcript:
1: so sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the
I think immediately after you play the next big revolution personally I think is an education and the complete homework for this
do you think ncrf plus any people together will be the next big revolution after up in India absolutely and why I think so is bec
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
July 2020 and we recently celebrated the 4th anniversary of NP 2020 ncrf has been brought to implement the intent of 2020 2020 th
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guide
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
pull up my kitchen to add to my taste buds in a way that it's convenient for me and one fine day you came and you changed my kitc
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will you be not only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we ar
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyin
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for no
it allows for creditor of all learnings weather in academics orange killing or an experiential learning and all these three kinds
and people who are already skilled or already in the professional area not there the experiential learning would play a big part
increase of technology which has been created by single Department good question so therefore all of us we have to work in tander
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills y
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of all
have you already seen the Fallout of this know you can see the photo you can see how many Engineers are there for Designing our b
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework ho
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept pract
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those ar
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 yea
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so th
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test t
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumin
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that w
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessme
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assessm
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and t
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you fe
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-minded
[Unintelligible]
play nursery and she says it lightly and the answer lies in this new framework and know how many times as you rightly say that st
it can be adopted but colleges that are approved by a city let's say a state Technical University may not know these rules of cou
of course a price them go and talk to them and then educate them but over and Beyond this there are some subtle problems that I a
play by 50% or I'll engage my faculty for the betterment of my students at a level which need not necessarily be teaching in the
picture Affiliated to these 1200 University leaving all the ionis that is Institute of national importance but they have about 4
2: 3 4 4
do you think this ratio is sufficient to teach a technical subject know once we are sending our students out again these teachers
it's not easy to create a project which is outcome based creating that project itself is going to take a lot of time similarly th
[Unintelligible]
challenge dead and then they're going to be no nutrition I suppose it is really against the students should not disincentivize re
what clothes should not be in the classroom but outside the classroom evaluation and teacher would be much happier teacher would
```

```
1 pip install sentence-transformers
```

```
RPL where are you you can get your learning validated if not having on Google on the course completely it feels like a hippie swa
Requirement already satisfied: sentence-transformers in /usr/local/lib/python3.11/dist-packages (2.6.1)
Requirement already satisfied: transformers, >=4.32.0, <5.0.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (2.5.1+cu124)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.13.1)
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (0.28
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from transformers, <5.0.0, >=4.32.0->sentence-trans
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (11.0.0)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-tra
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3.4
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3.1.5
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sent
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->se
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sent
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentenc
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->senten
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sent
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-tra
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-t
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->senten
Requirement already satisfied: triton==3.1.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers)
```

```
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch>=1.11.0->sen
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->sentence-tr
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->sente
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->sent
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->sentence-transformers)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->sentence-transfor
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch>=1.11.0->sentence-tran
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20.0->sent
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20.0
```

```
destroy world we can pick quickly they can pick quickly if they want to implement every solution is there in Seattle and if somel
single all possible handcuffs that education has converged to but with that comes a problem of a typical exhibition let's say like
tacos have been opened in a framework and that framework is very solid and unless that framework is underneath this will all be
at school level at college level at University level for Skilling Institution for ITI for polytechnics everywhere whether it is
number of creditors will have to find before and the assessment method and then you have to notify this again this part of the
can you sing happy songs I would say I will call for a hackathon of all the colleges in and around and I'll give them half a cree
enable from 120 our learning Center dome with small teachers we love going to the class teaching and coming out except that on
anything even on top of this is difficult for us because it's going to be time consuming for us and we also have a research mand
so far we cannot say that we have been getting best friends in education except for a few top institutions most of the people th
however there would be certain people who would like to work with them for them either here or in the industry anyway so we have
in the classrooms some minimize classrooms and increase volume
good morning assume I'm walking into a bachelor's program knowing good amount of physics it is BSN physics bsnbcs I don't want to
morning outcomes beforehand why should you be forced to go to the same course and waste your time why can't the time be used for
Integrity is very important that's an important parameter right that's the most important parameter here for the first time ugc
and this provides that the first semester the student cannot have that basic advantage of RPL he he he he cannot say that I w
```

```python
from sentence_transformers import SentenceTransformer
from sklearn.cluster import KMeans
import numpy as np
import nltk

# Load a pre-trained sentence embedding model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Load the transcript from a text file
input_file = "/content/Kalsi sir transcript.txt"  # Replace with your input file path
with open(input_file, "r") as file:
    transcript = file.read()

# Check if the transcript is empty
if not transcript.strip():
    raise ValueError("The input file is empty. Please provide a valid transcript.")

# Split the transcript into sentences using NLTK's sent_tokenize
nltk.download('punkt')
sentences = nltk.sent_tokenize(transcript)  # More robust sentence splitting
sentences = [s.strip() for s in sentences if s.strip()]  # Remove empty strings

# Debugging: Print the sentences to verify
print("Sentences extracted from the transcript:")
for i, sentence in enumerate(sentences):
    print(f"{i + 1}: {sentence}")

# Check if sentences are empty
if not sentences:
    raise ValueError("No sentences found in the transcript. Please check the input file.")

# Generate embeddings using Sentence Transformers
sentence_embeddings = model.encode(sentences)

# Apply K-Means clustering
num_clusters = 5  # Number of topics
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(sentence_embeddings)
cluster_labels = kmeans.labels_

# Group sentences by cluster
clustered_sentences = {i: [] for i in range(num_clusters)}
for sentence, label in zip(sentences, cluster_labels):
    clustered_sentences[label].append(sentence)

# Save the clusters to a file
output_file = "clustered_transcript.txt"  # Replace with your desired output file path
with open(output_file, "w") as f:
    for cluster_id, sentences_in_cluster in clustered_sentences.items():
        f.write(f"Cluster {cluster_id}:\n")
        for sentence in sentences_in_cluster:
            f.write(f" - {sentence}\n")
        f.write("\n")

print(f"Clustering complete. Output saved to '{output_file}'.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Sentences extracted from the transcript:
1: so sorry we know that India has seen a huge Revolution with digital payments we all thought that India is a place at least the
I think immediately after you play the next big revolution personally I think is an education and the complete homework for this
do you think ncrf plus any people together will be the next big revolution after up in India absolutely and why I think so is bec
how many changes have happened in the real world so many changes have happened in the requirement of the industry requirement of
July 2020 and we recently celebrated the 4th anniversary of NP 2020 ncrf has been brought to implement the intent of 2020 2020 th
this allows you all the Innovation the way you educate your kids you educate your students still it provides you the basic guide.
broken the shackles of which were there in the education sector right so yes it is a big Revolution and this is going to change t
pull up my kitchen to add to my taste buds in a way that it's convenient for me and one fine day you came and you changed my kitc
we have been driving buses at Max now we should fly how do we do this look at the requirement of the industry requirement of the
link that whatever he has been taught has no relevance to the real life world when industry is moving that fast when they require
will you be not only beneficial but also very facilitated for all of us very liberating for all of us this is going to be highly
new things create new ways of doing things learn something new but once we learn it there is no limit to Innovation and creativit
creating our vision and Innovative Minds into the education sector where is we are applying it elsewhere everywhere else no we ar
to be fun for everyone and I can I can tell you that already a number of Institutions have adopted the any pain and CRF to varyin
next step on how to implement an AP if I can request you to give me an elevator pitch for an EP and then an elevator pitch for nc
it allows for creditor of all learnings weather in academics orange killing or an experiential learning and all these three kinds
and people who are already skilled or already in the professional area not there the experiential learning would play a big part
increase of technology which has been created by single Department good question so therefore all of us we have to work in tander
all kinds of learning are being contractors including learning of soft skills employability skills life skills your hand skills y
go out it's all very flexible so that there's no Dropout there's no Dropout so these three things coupled with use of technology
I want my son to be an engineer don't you think if we create a give me five approach to Credit Systems everybody will come to the
turn off infrastructure India has today we created in last 1775 years we are going to double that infrastructure in next 10:11 ye
more number of other branches even liberal arts social sciences if I want to be an award Society I would need a proper mix of all
have you already seen the Fallout of this know you can see the photo you can see how many Engineers are there for Designing our b
[Unintelligible]
any of the new technology machines current any laser-based machines any automated operating machines robotic operations
robotic process and that is killing is equally important and this is important in multiple areas and therefore multidisciplinary
me and you want to know design a VTech in CSC syllabus or be taking AI syllabus that is Nip complaint under the ncrf framework ho
are you teaching Teddy teaching Terry is not sufficient if you want the student to really understand and reply that concept pract
those horses are skill bass courses you divide every subject into 30 and its application how do you apply the theory and those ar
who is learning which is happening which which you are going through so then this looks like let's say my student stays for 1 yea
extension of BSC physics or BSC chemistry and you give him the actual knowledge of computer science right in the first year so th
that's unbelievable so that 50% of the time which means two full years and a btech program a person can stay outside the compass
are you learning outcomes and their alignment with the overall curricular structure and then once it comes back we have to test t
he gets the credits okay so here is where I have talked with some inhibitions about the entire setup where you are simply assumin
write although we can keep that a check it is not easy for us to keep some zones green some zones red operational cost for that w
what is the guarantee that the student is learning in the campus is there a is there an accident on some kind of a giant assessme
either online or with some time stamps it is it is being documented know what time is done and CVT has videographed every assessm
she claims that all right he is a good technician and he can repair any kind of car so she prepares you open the BMW engine and t
Julie appointed by the awarding body which Awards the certificate and then that video is kept forever so you imagine won't you fe
who is giving us I think that's a nail on the head where I think we all should pass for a moment and then think is our education
experimenting something like this I don't think we'll be damaging I think all the Institute should come out of their close-minded
[Unintelligible]
play nursery and she says it lightly and the answer lies in this new framework and know how many times as you rightly say that st
it can be adopted but colleges that are approved by a city let's say a state Technical University may not know these rules of cou
of course a price them go and talk to them and then educate them but over and Beyond this there are some subtle problems that I n
play by 50% or I'll engage my faculty for the betterment of my students at a level which need not necessarily be teaching in the
picture Affiliated to these 1200 University leaving all the ionis that is Institute of national importance but they have about 4
2: 3 4 4
do you think this ratio is sufficient to teach a technical subject know once we are sending our students out again these teachers
it's not easy to create a project which is outcome based creating that project itself is going to take a lot of time similarly th
[Unintelligible]
challenge dead and then they're going to be no nutrition I suppose it is really against the students should not disincentivize re
what clothes should not be in the classroom but outside the classroom evaluation and teacher would be much happier teacher would
```

```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("WebOrganizer/TopicClassifier")
model = AutoModelForSequenceClassification.from_pretrained(
    "WebOrganizer/TopicClassifier",
    trust_remote_code=True,
    use_memory_efficient_attention=False
)

# Read the content from the uploaded text file
file_path = '/content/Kalsi sir transcript.txt'  # Replace with your actual file name
with open(file_path, 'r') as file:
    web_page = file.read()

# Tokenize the input text
inputs = tokenizer([web_page], return_tensors="pt")

# Get the model's predictions
outputs = model(**inputs)

# Compute the probabilities and get the predicted topic
probs = outputs.logits.softmax(dim=-1)
predicted_topic = probs.argmax(dim=-1).item()

# Print the predicted topic
print(f"Predicted Topic: {predicted_topic}")
# The predicted topic will correspond to the label (e.g., 5 for "Hardware")
```

```
have you been engaged with the best of the engineering colleges that has already been done and now we are also making a part of t
```

```
destroy world we can pick quickly they can pick quickly if they want to implement every solution is there in Seattle and if somel
tokenizer_config.json: 100%                                                    1.38k/1.38k [00:00<00:00, 34.0kB/s]
sing all possible hand                                                         t with that comes a problem of a typical exhibition let's say like
tacos have been opened in a framework and that framework is very solid and unless that framework is underneath this will all be s
vocab.txt: 100%                                                                232k/232k [00:00<00:00, 2.28MB/s]
at school lev                                                                  for skilling institution for ITI for polytechnics everywhere whether it is a
number of creditors will have to find before and the assessment method and then you have to notify this again this is part of the
tokenizer.json: 100%                                                           712k/712k [00:00<00:00, 5.10MB/s]
can you sing happ, songs a model say a kind case for a hackathon of all the colleges in and around and I'll give them half a cred
enable from 120 our Learni                                                     we love going to the class teaching and coming out except that or
special_tokens_map.json: 100%                                                  695/695 [00:00<00:00, 9.51kB/s]
anything over on top of this is difficult for us because it's going to be time-consuming for us and we also have a research manda
so far we comm                                                                 iends in education except for a few top institutions rest of the people the
config.json: 100%                                                              2.68k/2.68k [00:00<00:00, 36.6kB/s]
however there would be certain people who would like to work with them for them either here or in the industry anyway so we have
in the classroom                                                               olume
configuration.py: 100%                                                         7.13k/7.13k [00:00<00:00, 199kB/s]
good morning assume I'm walking into a bachelor's program knowing good amount of physics it is BSN physics bsnbcs I don't want to
morning outcomes beforehand why should you be forced to go to the same course and waste your time why can't the time be used for
- configuration.py
Integrity is very important that's an important parameter right that's the most important parameter here for the first time ugc i
  Make sure to double-check they do not contain any added malicious code. To avoid downloading new versions of the code file, you ca
and this provides that the first semester the student cannot have that basic advantage of RPL he he he he he cannot say that I w
day danger he car                                                              on my own or I've done it through self-study you tell me the learning out
course up to 50% of the time or up to 50% of the credits can be earned either through online courses or through RPM apprenticesh
in favor of the operation but as of today this is what I have been provided it is significant number of me things that I can affo
- modeling.py
institution who are not working around that they will teach the entire engineering to the student in the first one and a half yea
  Make sure to double-check they do not contain any added malicious code. To avoid downloading new versions of the code file, you ca
do students have to be placed in highly repeated companies like Google Microsoft article some of the top companies in the world a
model.safetensors: 100%                                                        550M/550M [00:06<00:00, 92.6MB/s]
play other students                                                            a half years very nice model anyone can afford model weather so a co
this is never integrated into the curriculum properly which I think is very important but sincerity model Dimensions being in pl
```

```
so we picked up the critical learning outcomes the generic learning outcomes which every student must gain as a graduate so The (
play level descriptors based on a particular soft skills you will get a particular level in the NCR that course will be related
```

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Load the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("WebOrganizer/TopicClassifier")
model = AutoModelForSequenceClassification.from_pretrained(
    "WebOrganizer/TopicClassifier",
    trust_remote_code=True,
    use_memory_efficient_attention=False
)

# Read the content from the uploaded text file
input_file_path = '/content/Kalsi sir transcript.txt'  # Replace with your actual file name
output_file_path = 'predicted_segments.txt'  # Output file name

# Open the input file and read its content
with open(input_file_path, 'r') as file:
    content = file.read()

# Split the content into segments (e.g., by paragraphs)
segments = content.split('\n\n')  # Adjust the delimiter as needed

# Process each segment and write the results to the output file
with open(output_file_path, 'w') as output_file:
    for segment in segments:
        if segment.strip():  # Skip empty segments
            # Tokenize the segment
            inputs = tokenizer([segment], return_tensors="pt")

            # Get the model's predictions
            outputs = model(**inputs)

            # Compute the probabilities and get the predicted topic
            probs = outputs.logits.softmax(dim=-1)
            predicted_topic = probs.argmax(dim=-1).item()

            # Write the segment and its predicted topic to the output file
            output_file.write(f"Segment:\n{segment}\n")
            output_file.write(f"Predicted Topic: {predicted_topic}\n")
            output_file.write("-" * 50 + "\n")  # Separator for readability

print(f"Predicted segments have been saved to {output_file_path}")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Predicted segments have been saved to predicted_segments.txt
```

```
import nltk
from nltk.tokenize import sent_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import AgglomerativeClustering
import re
import numpy as np

# Download necessary NLTK data (if you haven't already)
```

```
try:
    nltk.data.find("tokenizers/punkt")
except LookupError:
    nltk.download("punkt")


def segment_text(file_path, num_clusters=5, clustering_method='agglomerative'):
    """
    Segments a text file into sections based on topic clustering,
    using either K-Means or Agglomerative Clustering.

    Args:
        file_path (str): The path to the text file.
        num_clusters (int): The number of topic clusters to create.
        clustering_method (str):  'kmeans' or 'agglomerative' (default: 'agglomerative').

    Returns:
        dict: A dictionary where keys are topic labels (cluster numbers) and
              values are lists of sentences belonging to that topic.
    """

    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            text = file.read()
    except FileNotFoundError:
        return "Error: File not found."
    except Exception as e:
        return f"Error reading file: {e}"

    # 1. Sentence Tokenization
    sentences = sent_tokenize(text)

    # 2. Text Vectorization (TF-IDF)
    vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)  # Remove common English stop words
    tfidf_matrix = vectorizer.fit_transform(sentences)

    # 3. Clustering
    if clustering_method == 'kmeans':
        from sklearn.cluster import KMeans  # Import here to avoid unnecessary dependency if not used

        kmeans = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)  # Set random_state for reproducibility
        kmeans.fit(tfidf_matrix)
        clusters = kmeans.labels_

    elif clustering_method == 'agglomerative':
        # Convert sparse matrix to dense matrix
        tfidf_matrix_dense = tfidf_matrix.toarray()

        agglomerative = AgglomerativeClustering(n_clusters=num_clusters, linkage='ward')  # 'ward' linkage minimizes variance
        clusters = agglomerative.fit_predict(tfidf_matrix_dense)

    else:
        return "Error: Invalid clustering method. Choose 'kmeans' or 'agglomerative'."

    # 4. Segment Creation
    segmented_text = {}
    for i, cluster in enumerate(clusters):
        if cluster not in segmented_text:
            segmented_text[cluster] = []
        segmented_text[cluster].append(sentences[i])

    return segmented_text


def write_segmented_text_to_file(segmented_text, output_file="segmented_text.txt"):
    """
    Writes the segmented text to a file, with each topic as a section.

    Args:
        segmented_text (dict): The dictionary containing segmented text.
        output_file (str): The name of the output file (default: "segmented_text.txt").
    """
    try:
        with open(output_file, 'w', encoding='utf-8') as outfile:
            for cluster, sentences in segmented_text.items():
                outfile.write(f"Topic {cluster + 1}:\n")  # Topics are numbered starting from 1
                for sentence in sentences:
                    outfile.write(sentence + "\n")
                outfile.write("\n")  # Add a blank line between topics
        print(f"Segmented text written to {output_file}")
    except Exception as e:
        print(f"Error writing to file: {e}")
```

```


def clean_filename(filename):
    """
    Sanitizes a filename by removing or replacing invalid characters.

    Args:
        filename (str): The original filename.

    Returns:
        str: A cleaned filename.
    """
    # Replace spaces with underscores
    filename = filename.replace(" ", "_")
    # Remove any characters that are not alphanumeric, underscores, or periods
    filename = re.sub(r"[^a-zA-Z0-9_.]", "", filename)
    return filename


def main():
    file_path = input("Enter the path to the text file: ")
    num_topics = int(input("Enter the desired number of topics (clusters): "))  # Get the desired number of topics from the user

    clustering_method = input("Enter the clustering method ('kmeans' or 'agglomerative'): ").lower()  # Get the desired clustering

    segmented_data = segment_text(file_path, num_topics, clustering_method)

    if isinstance(segmented_data, str):  # Handle error messages from segment_text
        print(segmented_data)
        return

    # Extract filename from path for a more descriptive output filename
    file_name_base = file_path.split('/')[-1].split('.')[0]  # Get filename without extension
    cleaned_filename = clean_filename(file_name_base)
    output_file_name = f"{cleaned_filename}_segmented.txt"

    write_segmented_text_to_file(segmented_data, output_file_name)


if __name__ == "__main__":
    main()

```

```
Enter the path to the text file: /content/Kalsi sir transcript.txt
Enter the desired number of topics (clusters): 2
Enter the clustering method ('kmeans' or 'agglomerative'): agglomerative
Segmented text written to Kalsi_sir_transcript_segmented.txt
```

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import networkx as nx

# Load and preprocess the text file
file_path = '/content/Kalsi sir transcript.txt'  # Replace with your text file pat
with open(file_path, 'r', encoding='utf-8') as file:
    paragraphs = file.read().split('\n\n')

# Convert paragraphs to TF-IDF vectors
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(paragraphs)

# Compute cosine similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix)

# Create a graph based on similarity thresholds
csim_threshold = 0.2  # Compare similarity value
graph = nx.Graph()
for i in range(len(paragraphs)):
    for j in range(i + 1, len(paragraphs)):
        if cosine_sim[i, j] > csim_threshold:
            graph.add_edge(i, j, weight=cosine_sim[i, j])

# Detect themes using connected components
themes = [list(component) for component in nx.connected_components(graph)]

# Generate a summary based on themes
summary = []
for theme in themes:
    theme_text = " ".join([paragraphs[i] for i in theme])
    summary.append(theme_text)

```

```
# Display themes and summary
for idx, theme in enumerate(summary):
    print(f"Theme {idx + 1}:\n{theme[:500]}...\n{'-'*50}")

```

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import networkx as nx

# Load and preprocess the text file
file_path = '/content/Kalsi sir transcript.txt'  # Replace with your text file path
with open(file_path, 'r', encoding='utf-8') as file:
    paragraphs = [p.strip() for p in file.read().split('\n\n') if p.strip()]

# Convert paragraphs to TF-IDF vectors
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(paragraphs)

# Compute cosine similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix)

# Create a graph based on similarity thresholds
csim_threshold = 0.1  # Compare similarity value
graph = nx.Graph()
for i in range(len(paragraphs)):
    for j in range(i + 1, len(paragraphs)):
        if cosine_sim[i, j] > csim_threshold:
            graph.add_edge(i, j, weight=cosine_sim[i, j])

# Detect themes using connected components
themes = [list(component) for component in nx.connected_components(graph)]

# Generate a summary based on themes
summary = []
for theme in themes:
    # Extract a representative paragraph for each theme (first paragraph in the theme)
    theme_text = " ".join([paragraphs[i] for i in theme])
    summary.append(theme_text)

# Save themes to a text file
with open('themes_output.txt', 'w', encoding='utf-8') as out_file:
    for idx, theme in enumerate(summary):
        out_file.write(f"Theme {idx + 1}:\n{theme}\n{'-'*50}\n")

# Print a summary of themes
print(f"Detected {len(themes)} themes. Check 'themes_output.txt' for details.")
for idx, theme in enumerate(summary):
    print(f"Theme {idx + 1} (Preview):\n{theme[:300]}...\n{'-'*50}")

```

```
Detected 0 themes. Check 'themes_output.txt' for details.
```

```
print("Cosine Similarity Matrix:\n", cosine_sim)
```

```
Cosine Similarity Matrix:
 [[1.]]
```

```
print(f"Graph Info: Nodes = {graph.number_of_nodes()}, Edges = {graph.number_of_edges()}")
```

```
Graph Info: Nodes = 0, Edges = 0
```

```
pip install networkx
```

```
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (3.4.2)
```

```
from transformers import pipeline

def segment_transcript_local(text):
    """Uses a local LLM (Hugging Face) to segment the transcript into meaningful topics."""

    summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

    chunks = [text[i:i+1024] for i in range(0, len(text), 1024)]
```