DO NOT RUN ALL THE CODES ON COLAB

```
pip install yt-dlp
```

```
Collecting yt-dlp
  Downloading yt_dlp-2025.3.31-py3-none-any.whl.metadata (172 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 172.2/172.2 kB 2.2 MB/s eta 0:00:00
Downloading yt_dlp-2025.3.31-py3-none-any.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 81.0 MB/s eta 0:00:00
Installing collected packages: yt-dlp
Successfully installed yt-dlp-2025.3.31
```

Download Library to automate cookie downloading

USE VS Code

```
pip install selenium
```

```
Collecting selenium
  Downloading selenium-4.31.0-py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: urllib3<3,>=1.26 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26->selenium)
Collecting trio~=0.17 (from selenium)
  Downloading trio-0.29.0-py3-none-any.whl.metadata (8.5 kB)
Collecting trio-websocket~=0.9 (from selenium)
  Downloading trio_websocket-0.12.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: certifi>=2021.10.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (2025.1.31)
Requirement already satisfied: typing_extensions~=4.9 in /usr/local/lib/python3.11/dist-packages (from selenium) (4.13.2)
Requirement already satisfied: websocket-client~=1.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (1.8.0)
Requirement already satisfied: attrs>=23.2.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (25.3.0)
Requirement already satisfied: sortedcontainers in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (3.10)
Collecting outcome (from trio~=0.17->selenium)
  Downloading outcome-1.3.0.post0-py2.py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: sniffio>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (1.3.1)
Collecting wsproto>=0.14 (from trio-websocket~=0.9->selenium)
  Downloading wsproto-1.2.0-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26
Requirement already satisfied: h11<1,>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from wsproto>=0.14->trio-websocket~=0.9->se
Downloading selenium-4.31.0-py3-none-any.whl (9.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 127.2 MB/s eta 0:00:00
Downloading trio-0.29.0-py3-none-any.whl (492 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 492.9/492.9 kB 37.8 MB/s eta 0:00:00
Downloading trio_websocket-0.12.2-py3-none-any.whl (21 kB)
Downloading outcome-1.3.0.post0-py2.py3-none-any.whl (10 kB)
Downloading wsproto-1.2.0-py3-none-any.whl (24 kB)
Installing collected packages: wsproto, outcome, trio, trio-websocket, selenium
Successfully installed outcome-1.3.0.post0 selenium-4.31.0 trio-0.29.0 trio-websocket-0.12.2 wsproto-1.2.0
```

```
pip install undetected-chromedriver
```

```
Collecting undetected-chromedriver
  Downloading undetected-chromedriver-3.5.5.tar.gz (65 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.4/65.4 kB 6.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: selenium>=4.9.0 in /usr/local/lib/python3.11/dist-packages (from undetected-chromedriver) (4.31.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from undetected-chromedriver) (2.32.3)
Requirement already satisfied: websockets in /usr/local/lib/python3.11/dist-packages (from undetected-chromedriver) (15.0.1)
Requirement already satisfied: urllib3<3,>=1.26 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26->selenium>=
Requirement already satisfied: trio~=0.17 in /usr/local/lib/python3.11/dist-packages (from selenium>=4.9.0->undetected-chromedriver
Requirement already satisfied: trio-websocket~=0.9 in /usr/local/lib/python3.11/dist-packages (from selenium>=4.9.0->undetected-chro
Requirement already satisfied: certifi>=2021.10.8 in /usr/local/lib/python3.11/dist-packages (from selenium>=4.9.0->undetected-chrom
Requirement already satisfied: typing_extensions~=4.9 in /usr/local/lib/python3.11/dist-packages (from selenium>=4.9.0->undetected-c
Requirement already satisfied: websocket-client~=1.8 in /usr/local/lib/python3.11/dist-packages (from selenium>=4.9.0->undetected-ch
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->undetected-chrome
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->undetected-chromedriver) (3.1
Requirement already satisfied: attrs>=23.2.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium>=4.9.0->undetecte
Requirement already satisfied: sortedcontainers in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium>=4.9.0->undete
Requirement already satisfied: outcome in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium>=4.9.0->undetected-chro
Requirement already satisfied: sniffio>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium>=4.9.0->undetect
Requirement already satisfied: wsproto>=0.14 in /usr/local/lib/python3.11/dist-packages (from trio-websocket~=0.9->selenium>=4.9.0-
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26
Requirement already satisfied: h11<1,>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from wsproto>=0.14->trio-websocket~=0.9->se
Building wheels for collected packages: undetected-chromedriver
  Building wheel for undetected-chromedriver (setup.py) ... done
  Created wheel for undetected-chromedriver: filename=undetected_chromedriver-3.5.5-py3-none-any.whl size=47047 sha256=145fe0eb074a3
  Stored in directory: /root/.cache/pip/wheels/5c/b9/03/4b6e38f019d6170e8c25df2e1e362d7bdf9ff4012df2dc85c0
Successfully built undetected-chromedriver
Installing collected packages: undetected-chromedriver
Successfully installed undetected-chromedriver-3.5.5
```

```
pip install webdriver-manager
```

```
Collecting webdriver-manager
  Downloading webdriver_manager-4.0.2-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from webdriver-manager) (2.32.3)
Collecting python-dotenv (from webdriver-manager)
  Downloading python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from webdriver-manager) (24.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->webdriver-manager
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->webdriver-manager) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->webdriver-manager) (2.3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->webdriver-manager) (202
Downloading webdriver_manager-4.0.2-py2.py3-none-any.whl (27 kB)
Downloading python_dotenv-1.1.0-py3-none-any.whl (20 kB)
Installing collected packages: python-dotenv, webdriver-manager
Successfully installed python-dotenv-1.1.0 webdriver-manager-4.0.2
```

```
pip install setuptools
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (75.2.0)
```

Video to Audio generation

[ ] ↳ 4 cells hidden

Audio to Transcript Generation

[ ] ↳ 2 cells hidden

Segmenting Transcripts based on the contexts and topic

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import spacy
import numpy as np
from scipy.signal import find_peaks
import random

# Set random seed for reproducibility
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed_all(SEED)

# Load spaCy tokenizer
nlp = spacy.load("en_core_web_sm")

def load_text_file(file_path):
    """Load transcript with sentence-level timestamps."""
    sentences = []
    timestamps = []

    with open(file_path, 'r', encoding='utf-8') as f:
        lines = f.readlines()

    for i in range(0, len(lines), 3):
      if i + 1 >= len(lines):  # Skip if not enough lines left
          continue

      time_range = lines[i].strip().split(" --> ")

      # Skip invalid time ranges
      if len(time_range) != 2:
          continue

      try:
          start_time = float(time_range[0])
          end_time = float(time_range[1])
      except ValueError:
```

```
          continue

      text = lines[i + 1].strip()

      if text:  # Only add if text is not empty
          sentences.append(text)
          timestamps.append((start_time, end_time))

    tokens = [token.text for sent in sentences for token in nlp(sent)]
    return sentences, tokens, timestamps


class Encoder(nn.Module):
    def __init__(self, input_dim, hidden_dim):
        super(Encoder, self).__init__()
        self.hidden_dim = hidden_dim
        self.bigru = nn.GRU(input_dim, hidden_dim, bidirectional=True, batch_first=True)

    def forward(self, x):
        h, _ = self.bigru(x)
        return h  # h ∈ R^(N × 2H)


class Decoder(nn.Module):
    def __init__(self, hidden_dim):
        super(Decoder, self).__init__()
        self.hidden_dim = hidden_dim
        self.gru = nn.GRU(hidden_dim * 2, hidden_dim, batch_first=True)

    def forward(self, x, hidden_state):
        d, hidden_state = self.gru(x, hidden_state)
        return d, hidden_state


class Pointer(nn.Module):
    def __init__(self, encoder_hidden_dim, decoder_hidden_dim):
        super(Pointer, self).__init__()
        self.W1 = nn.Linear(encoder_hidden_dim, decoder_hidden_dim)
        self.W2 = nn.Linear(decoder_hidden_dim, decoder_hidden_dim)
        self.v = nn.Linear(decoder_hidden_dim, 1, bias=False)

    def forward(self, encoder_outputs, decoder_state):
        scores = self.v(torch.tanh(self.W1(encoder_outputs) + self.W2(decoder_state)))
        attention_weights = F.softmax(scores, dim=1)
        return attention_weights


class SEGBOT(nn.Module):
    def __init__(self, input_dim, hidden_dim):
        super(SEGBOT, self).__init__()
        self.encoder = Encoder(input_dim, hidden_dim)
        self.decoder = Decoder(hidden_dim)
        self.pointer = Pointer(hidden_dim * 2, hidden_dim)

    def forward(self, x, start_units):
        encoder_outputs = self.encoder(x)
        decoder_hidden = torch.zeros(1, x.size(0), self.decoder.hidden_dim).to(x.device)
        decoder_inputs = encoder_outputs[:, start_units, :].unsqueeze(1)
        decoder_outputs, _ = self.decoder(decoder_inputs, decoder_hidden)
        attention_weights = self.pointer(encoder_outputs, decoder_outputs.squeeze(1))
        return attention_weights

    def segment_text(self, sentences, tokens, timestamps, attention_weights):
        """Segment text and get start/end timestamps."""
        attention_weights = attention_weights.squeeze().detach().cpu().numpy()

        # Normalize attention weights
        attention_weights = (attention_weights - np.min(attention_weights)) / (
            np.max(attention_weights) - np.min(attention_weights)
        )

        # Find peaks in attention scores
        peak_indices, _ = find_peaks(attention_weights, height=0.5, distance=5)

        if len(peak_indices) == 0:
            return [{"text": " ".join(sentences), "start_time": timestamps[0][0], "end_time": timestamps[-1][1]}]

        segments = []
        start_idx = 0
        for i in peak_indices:
          if i > 0 and i - start_idx >= 5:  # Ensure valid range and at least 5 sentences per segment
              segment_text = " ".join(sentences[start_idx:i]).strip()
```

```
123
124             if segment_text:
125                 start_time = timestamps[start_idx][0]
126
127                 # Check if `i - 1` is within range to prevent out-of-bounds error
128                 if i - 1 < len(timestamps):
129                     end_time = timestamps[i - 1][1]
130                 else:
131                     end_time = timestamps[-1][1]  # Fallback to the last timestamp
132
133                 segments.append({"text": segment_text, "start_time": start_time, "end_time": end_time})
134
135             start_idx = i
136
137
138         # Add last segment
139         last_segment = " ".join(sentences[start_idx:]).strip()
140         if last_segment:
141             start_time = timestamps[start_idx][0]
142             end_time = timestamps[-1][1]
143             segments.append({"text": last_segment, "start_time": start_time, "end_time": end_time})
144
145         return segments if segments else None
146
147
148 # Model Hyperparameters
149 input_dim = 128  # Example input size
150 hidden_dim = 256  # Hidden layer size
151 model = SEGBOT(input_dim, hidden_dim)
152
153 # Load text file and process with sentence-level timestamps
154 file_path = "/content/sentence_timestamps.txt"  # Ensure sentence-level transcript format
155 sentences, tokens, timestamps = load_text_file(file_path)
156
157 # Example Input (Dummy Tensor)
158 x = torch.randn(1, len(tokens), input_dim)  # Batch size of 1, sequence length based on text
159 start_units = 0
160 output = model(x, start_units)
161
162 # Segment the text and get timestamps
163 segments = model.segment_text(sentences, tokens, timestamps, output)
164
165 # Save segmented transcript with timestamps
166 if segments:
167     with open("segmented_transcript_with_timestamps.txt", "w", encoding="utf-8") as f:
168         for i, segment in enumerate(segments):
169             start_time = segment["start_time"]
170             end_time = segment["end_time"]
171             text = segment["text"]
172             f.write(f"Segment {i+1} [{start_time:.2f}s - {end_time:.2f}s]:\n{text}\n\n")
173     print("Segmented transcript with timestamps saved successfully.")
174 else:
175     print("No valid segments found. Terminating execution.")
176
```