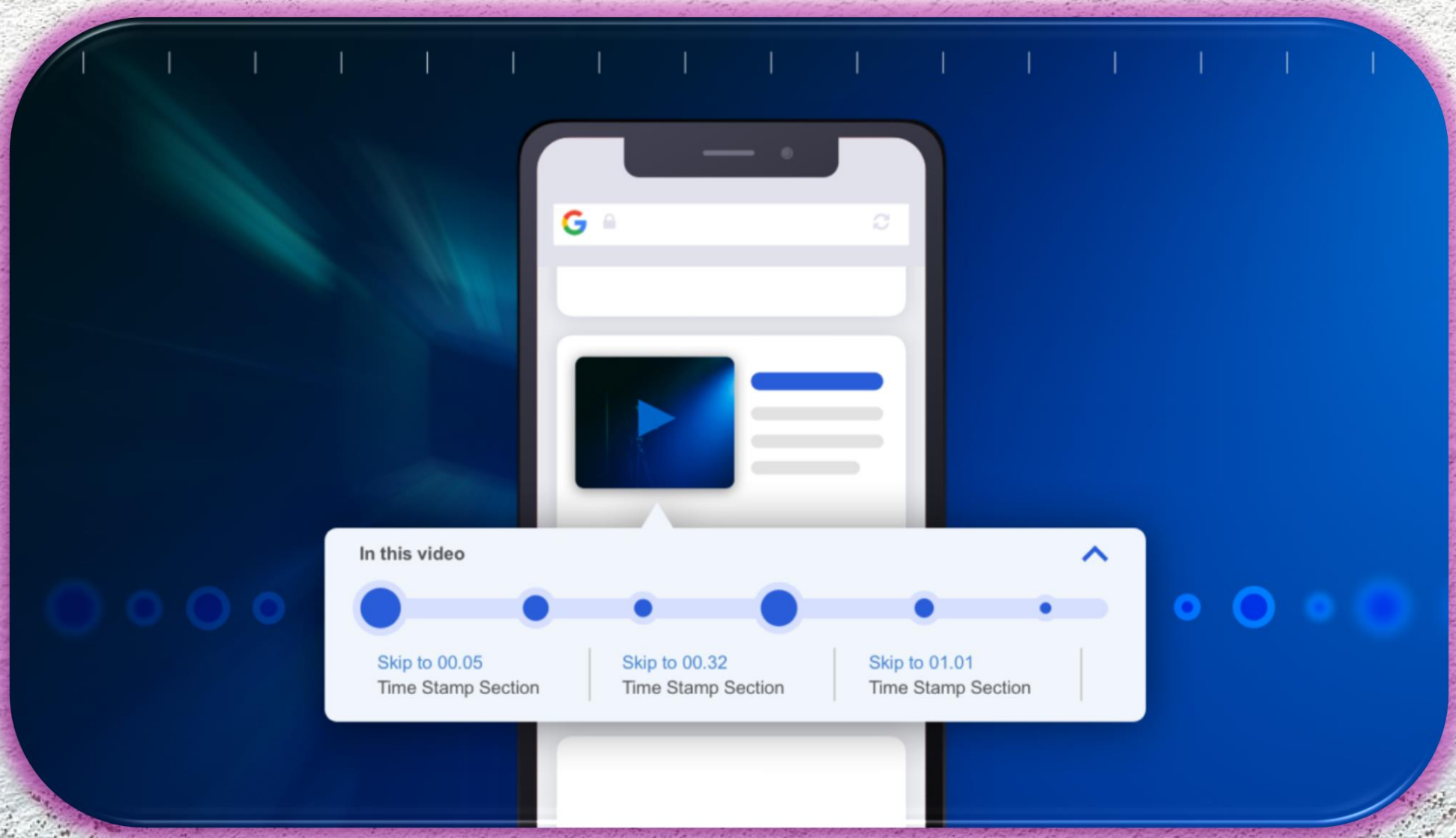
Automatically Generating Timestamps for Videos Using

OpenAI



ANSHUMAN JHA

Table of Contents

1. Introduction

In Imagine you've just recorded an hour-long interview or tutorial, and now you want to make it easier for viewers to skip to the sections they care about most. Manually generating timestamps can be a tedious task, especially for longer videos.

This is where OpenAI's models come to the rescue. With tools like Whisper (for transcription) and GPT (for summarization), you can automatically create timestamps that break down your video into digestible chunks, highlighting the key points of your content.

Let's dive into how this works, and look at some real-world examples along the way.

2. Why Do You Need Timestamps?

Timestamps act as a roadmap for your video, allowing viewers to easily navigate to the parts they find most interesting. For example:

In a cooking video, viewers might want to skip to the part where you're assembling the ingredients.

In a lecture or presentation, timestamps can break down the video into sections like "Introduction," "Main Concepts," and "Q&A."

In a podcast, timestamps can highlight when different guests speak or when specific topics are discussed.

By using OpenAI's technology, you can automate this process, making your content more accessible and user-friendly without the hours of manual work.

3. Step 1: Extracting Audio from the Video

Before diving into transcription, we need to extract the audio from the video.

Why? Because tools like Whisper, which we'll use for transcription, work best with audio files.

Let's say you're a content creator who just recorded a two-hour-long video podcast.

The first step would be to extract the audio from your video.

You can use free tools like moviepy, which can handle video files and extract the audio portion. Once you have the audio file, we're ready for the next step.

4.Step 2: Transcribing the Audio with Whisper

Now comes the magic! OpenAI's Whisper is an advanced speech recognition model that can transcribe your audio into text.

The transcription process turns every spoken word in your video into written text, giving you a complete script of what was said.

For instance, if you're a YouTuber who's just published a tutorial on web development, Whisper can listen to the entire video and transcribe all of your explanations, step-by-step.

This is especially useful for:

- ***Interviews, where each person's speech is captured.***
- ***Lectures, where a professor's explanation can be transcribed verbatim.***
- ***Podcasts, where Whisper can generate transcripts that enhance accessibility for your audience***

Whisper is incredibly accurate, even with different accents or background noise, making it ideal for a wide variety of content.

4.1 Example: You Just Recorded a 40-Minute Workout Video

You can use Whisper to transcribe your fitness video, breaking down every segment into text. For example:

. Introduction to the workout: "Hey everyone, today we're focusing on core exercises."

. Main exercises: "Let's start with crunches for 30 seconds..."

Now that we have the full transcription, we can move on to summarizing the content into actionable timestamps.

5. Step 3: Summarizing the Transcript into Timestamps

Having a full transcription is great, but it's not enough for creating useful timestamps.

We need to break down the transcript into sections that viewers can quickly jump to.

This is where OpenAI's GPT model comes in. GPT can analyze chunks of your transcription and summarize key points into meaningful sections.

For instance, instead of reading through an entire 40-minute fitness tutorial, GPT can automatically identify when you switch from "Warm-up" to "Core Exercises" to "Cool Down."

Imagine you just recorded a business webinar on marketing strategies:

GPT can summarize the transcript and identify key sections like:

- *00:00 - 05:00: Introduction to digital marketing trends.*
- *05:01 - 20:00: Step-by-step guide to optimizing your social media.*
- *20:01 - 40:00: Live Q&A with the audience.*

6. Step 4: Associating Summaries with Timestamps

When Whisper transcribes the audio, it not only gives you the text but also includes timestamps for when each word or sentence was spoken.

We can now combine these timestamps with the summaries generated by GPT.

For example, in your 40-minute workout video, GPT might generate summaries like this:

- *00:00 - 01:30: Introduction and warm-up.*
- *01:31 - 10:00: Core strengthening exercises.*
- *10:01 - 20:00: Cardio interval training.*

This means you can display the summaries as clickable timestamps in the video description, allowing viewers to jump directly to the segments they're most interested in.

7.Step 5: Formatting and Displaying the Timestamps

Finally, we format the timestamps into a viewer-friendly style.

Instead of using raw numbers like "90 seconds," we convert them into minutes and seconds (01:30 for 1 minute 30 seconds).

This makes the timestamps easy to read and navigate.

Example: A Cooking Show Episode

Let's say you just uploaded a 25-minute cooking video where you show how to make lasagna. Here's what your automatically generated timestamps might look like:

- ***00:00 - 02:30: Introduction and ingredients.***
- ***02:31 - 10:00: Preparing the sauce.***
- ***10:01 - 18:00: Assembling the lasagna layers.***
- ***18:01 - 25:00: Baking and final presentation.***

Viewers can click these timestamps and skip directly to the parts they care about, whether it's the sauce preparation or final dish reveal.

8. Benefits of Automating Timestamp Generation

. ***Saves Time:*** *For content creators, automating timestamp creation frees up time that would otherwise be spent manually scrubbing through videos.*

. ***Enhances User Experience:*** *Timestamps help viewers quickly navigate to the sections they're most interested in, improving overall engagement.*

. ***Increases Accessibility:*** *By providing a transcript and timestamps, your content becomes more accessible to a wider audience, including those with hearing impairments or those who prefer reading over watching.*

9. Limitations and Challenges

While OpenAI's tools are powerful, they're not perfect. Some limitations and challenges you may face include:

- **Accuracy:** Whisper may struggle with heavy accents, background noise, or technical jargon, leading to inaccuracies in the transcription.
- **Language Support:** Whisper and GPT are primarily trained on English content, so their performance may vary when working with other languages.

- ***Cost:*** ***Using OpenAI's API can be expensive, especially for longer videos or high-volume transcription needs.***

To mitigate these challenges, it's essential to review and edit the automatically generated timestamps to ensure accuracy and clarity. You may also need to fine-tune the models on your specific domain or language for better results.

10. Integrating Timestamps into Video Platforms

Once you have your generated timestamps, you can easily integrate them into various video platforms:

- *YouTube: Add the timestamps to your video description, following the format MM:SS - Description.*

- *Vimeo: Use the "Custom Sections" feature to add timestamps and descriptions to your video.*

- ***Wistia:*** ***Use the "Chapters" feature to add timestamps and titles to your video.***

By integrating timestamps into your video platform of choice, you make it easy for viewers to navigate your content and find the sections they're most interested in.

11. Conclusion

OpenAI's Whisper and GPT models make it incredibly easy to generate timestamps for any video.

Whether you're a content creator, educator, or marketer, automating this process can help save time and enhance the overall viewer experience.

From cooking shows to webinars and fitness tutorials, this method allows you to summarize key moments and present them in an easy-to-digest format for your audience.

So, the next time you're uploading a video, why not let AI do the hard work of timestamping for you?

You'll make your content more engaging and accessible while saving hours of manual labor.

Example Google Colab Notebook

Automatically_Generating_Timestamps_for_Videos_Using_OpenAI

September 8, 2024

1 Automatically Generating Timestamps for Videos Using OpenAI

In this tutorial, we'll explore how to automatically generate timestamps for videos using OpenAI's capabilities. This will involve breaking down the video's content, transcribing the audio, and segmenting it into meaningful timestamps. We'll use OpenAI's GPT model to assist in summarizing the content into sections that can serve as chapter markers or timestamps. We'll walk through a Google Colab notebook where we can implement this step by step.

Prerequisites:

- OpenAI API Key
- Google Colab (or local Jupyter Notebook)
- Python knowledge
- A video file (or audio file) to work with

1.0.1 Step 1: Setting Up the Environment

First, we need to set up the necessary libraries in Google Colab. We will use libraries for transcription (such as `whisper`), `moviepy` for handling video, and `openai` for text processing.

Install necessary libraries:

[ ]:
```
!pip install git+https://github.com/openai/whisper.git
```

[ ]:
```
!pip install openai==0.28 moviepy
```

1.0.2 Step 2: Upload Video and Extract Audio

For transcription, we'll extract the audio from the video using the `moviepy` library.

[ ]:
```
from moviepy.editor import VideoFileClip

# Path to the video file
video_path = "/content/test-video.mp4"

# Load the video
```

```
video_clip = VideoFileClip(video_path)

# Extract the audio and save it as a separate file
audio_path = "extracted_audio.wav"
video_clip.audio.write_audiofile(audio_path)
```

```
MoviePy - Writing audio in extracted_audio.wav


MoviePy - Done.
```

1.0.3 Step 3: Transcribe the Audio Using Whisper

We will use the `whisper` model to transcribe the audio into text. Whisper is an automatic speech recognition (ASR) model developed by OpenAI.

Transcription with Whisper:

[ ]:
```
import whisper

# Load the Whisper model
model = whisper.load_model("base")

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

# Extract the transcript
transcript = result['text']
timestamps = result['segments']  # To capture timestamps for each segment
```

```
WARNING:py.warnings:/usr/local/lib/python3.10/dist-
packages/whisper/__init__.py:146: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the default pickle
module implicitly. It is possible to construct malicious pickle data which will
execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  checkpoint = torch.load(fp, map_location=device)

WARNING:py.warnings:/usr/local/lib/python3.10/dist-
```

```
packages/whisper/transcribe.py:126: UserWarning: FP16 is not supported on CPU;
using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
```

Explanation:

- `whisper.load_model('base')`: Loads the small ASR model from Whisper, ideal for moderate-sized transcription tasks.
- `model.transcribe(audio_path)`: Transcribes the audio and returns a dictionary containing the text and other metadata (like timestamps).

1.0.4 Step 4: Break the Transcript into Meaningful Segments

After we obtain the transcription, the next step is to split it into logical sections (e.g., every few minutes or based on topic shifts). OpenAI GPT models can help summarize these sections or group the transcript into different topics.

Function to Generate Summaries:

[ ]:
```python
import openai

# OpenAI API key setup
openai.api_key = "your_OpenAI_API_key"

def summarize_text(text):
    response = openai.ChatCompletion.create(
      model="gpt-3.5-turbo",
      messages=[
        {"role": "system", "content": "You are an expert at summarizing text␣
 ↪into distinct topics."},
        {"role": "user", "content": f"Summarize the following text into␣
 ↪distinct topics:\n\n{text}"}
      ],
      max_tokens=150,
      temperature=0.5,
      top_p=1,
      frequency_penalty=0,
      presence_penalty=0
    )

    summary = response['choices'][0]['message']['content'].strip()
    return summary
```

Explanation:

- `openai.Completion.create`: Uses GPT-3.5-turbo to generate summaries of the text. The `prompt` instructs the model to split the transcript into meaningful sections.

1.0.5 Step 5: Associate Summaries with Timestamps

We can now associate the generated summaries with the timestamps we obtained from Whisper. This will allow us to create timestamps for each significant section of the video.

[ ]:
```
def generate_timestamps(transcript, segments):
    timestamped_sections = []

    for segment in segments:
        start_time = segment['start']
        end_time = segment['end']
        text = segment['text']

        # Generate summary for this segment
        summary = summarize_text(text)

        # Append to the list
        timestamped_sections.append({
            "start": start_time,
            "end": end_time,
            "summary": summary
        })

    return timestamped_sections
```

Explanation:

- `segments`: These are the timestamps from Whisper.
- For each segment, we generate a summary and associate it with the start and end times of that section.

1.0.6 Step 6: Display Timestamps in Readable Format

We can format and display the generated timestamps in a human-readable format, such as `MM:SS`.

[ ]:
```
import math

def seconds_to_time(seconds):
    minutes = math.floor(seconds / 60)
    seconds = round(seconds % 60)  # Round seconds to avoid float issues
    return f"{minutes:02d}:{seconds:02d}"  # Now it's safe to format seconds as␣
 ↪an integer

def display_timestamps(timestamped_sections):
    for section in timestamped_sections:
        start_time = seconds_to_time(section['start'])
        end_time = seconds_to_time(section['end'])
        summary = section['summary']
```

```
        print(f"{start_time} - {end_time}: {summary}\n")
```

Explanation:

- `seconds_to_time`: Converts seconds into `MM:SS` format.
- `display_timestamps`: Loops through each timestamped section and prints the start/end times along with the summary.

1.0.7 Step 7: Putting It All Together

Here's how we can integrate everything into a single function that extracts timestamps from a video, transcribes it, summarizes it, and displays the results.

```
[ ]: def generate_video_timestamps(video_path):
         # Extract audio from the video
         video_clip = VideoFileClip(video_path)
         audio_path = "extracted_audio.wav"
         video_clip.audio.write_audiofile(audio_path)

         # Transcribe the audio
         model = whisper.load_model("base")
         result = model.transcribe(audio_path)
         transcript = result['text']
         segments = result['segments']

         # Generate summaries with timestamps
         timestamped_sections = generate_timestamps(transcript, segments)

         # Display the timestamped summaries
         display_timestamps(timestamped_sections)
```

```
[ ]: # Test the function
     generate_video_timestamps("/content/test-video.mp4")
```

```
MoviePy - Writing audio in extracted_audio.wav


MoviePy - Done.

WARNING:py.warnings:/usr/local/lib/python3.10/dist-
packages/whisper/__init__.py:146: FutureWarning: You are using `torch.load` with
`weights_only=False` (the current default value), which uses the default pickle
module implicitly. It is possible to construct malicious pickle data which will
execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
```

```
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  checkpoint = torch.load(fp, map_location=device)

WARNING:py.warnings:/usr/local/lib/python3.10/dist-
packages/whisper/transcribe.py:126: UserWarning: FP16 is not supported on CPU;
using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")


00:00 - 00:08: 1. Repelit agent: This topic covers the concept of using Repelit
agent as the easiest way to create, debug, and prototype applications or
websites. It may include information on the features and benefits of using
Repelit agent for development purposes.

00:08 - 00:15: 1. New feature introduction
2. Planning and strategizing
3. Progress monitoring
4. Testing and deployment

00:15 - 00:19: 1. Stock data visualization app creation
2. Rapid development in a few minutes

00:19 - 00:25: 1. Stock symbol selection
2. Data fetching and retrieval

00:25 - 00:29: The text seems to mention two distinct topics:
1. Graphs: The text refers to something related to graphs, but it is not clear
what specific aspect of graphs is being discussed.
2. Downloading in CS reformat: The text mentions the ability to download
something in a CS (Computer Science) reformat, implying a process or method
related to downloading files in a specific format.

00:29 - 00:34: 1. Map creation
2. Nearest landmark display

00:34 - 00:36: The text can be summarized into one distinct topic: quick
creation or efficiency in completing a task.

00:36 - 00:42: 1. Request for App Development
2. Map Display
3. Local Landmarks

00:42 - 00:45: The text appears to be discussing the concept of variable areas
on a map. The distinct topics that can be summarized from this text are:
```

1. Variable areas on maps: This topic involves the idea that different areas on a map can have varying sizes or scales, which may be represented through different visual techniques or data visualization methods.

2. Mapping techniques: This topic could cover the various methods and tools used to represent variable areas on maps, such as cartographic techniques, GIS software, or data visualization platforms.

3. Spatial analysis: This topic may involve the examination and interpretation of data within variable areas on a map to derive insights, patterns, or relationships, which can be useful for decision-making or planning purposes.

00:45 - 00:49: 1. Utilizing Wikipedia for obtaining information on landmarks
2. Following steps for a specific purpose

00:49 - 00:52: The text can be summarized into one distinct topic:
- Previewing the app directly on the web page

00:52 - 00:57: 1. Progress tracking
2. Automation of file creation

00:57 - 00:59: Topic 1: Installation process
- Installing dependencies

This text primarily discusses the process of installing dependencies for a certain task or project.

00:59 - 01:00: The text provided is very brief and lacks specific details to summarize into distinct topics.

01:00 - 01:06: Topic 1: Integration with database as a default feature.

01:06 - 01:15: 1. Postgres SQL
2. Integrations with Google Docs
3. Integrations with Slack
4. Integrations with Discord
5. Integrations with Stripe
6. Other integrations

01:15 - 01:21: 1. AI agent capabilities: The text discusses how the AI agent has the ability to create code and integrate with applications.
2. Ease of use: It highlights how the AI agent makes the process easy for users to work with applications.

01:21 - 01:25: 1. Data Visualization App
2. Creation Process

01:25 - 01:28: The text can be summarized into the following distinct topics:
1. Natural language prompts
2. Expertise in summarizing text

01:28 - 01:32: 1. News app development
2. Scraper implementation

01:32 - 01:36: 1. Scraper tool
2. Hacker News website
3. Continuous scraping operation

01:36 - 01:39: The text can be summarized into the following distinct topics:
1. Data collection
2. Database storage

01:39 - 01:43: 1. News app development
2. Front end design

01:43 - 01:46: The text does not provide enough information to identify distinct topics.

01:46 - 01:51: 1. Creation of videos on YouTube channel
2. Cost of artificial intelligence

01:51 - 01:53: The text can be summarized into the following distinct topics:
1. Subscription
2. Bell icon
3. Staying tuned

01:53 - 01:57: 1. Call to action: Encouraging viewers to click the like button
2. Social sharing: Promoting the video to help others

01:57 - 01:59: The text is too brief to be summarized into distinct topics.

01:59 - 02:05: 1. Database: Postgres database
2. Front end: Next JS

02:05 - 02:07: Topic 1: Signing up for Replit

02:07 - 02:13: 1. AI agent feature
2. Availability in Replit, Core, and Teams

02:13 - 02:16: The text is too brief to identify distinct topics.

02:16 - 02:21: 1. App Development: The text discusses the desire to create a news app.
2. Data Collection: The app will continuously scrape news from news.ycompinated.com.

02:21 - 02:27: 1. Data population in Postgres database
2. Rendering news in Next.js frontend

02:27 - 02:29: I'm sorry, but there is no text provided for me to summarize into distinct topics. If you provide me with specific text or content, I would be happy to help summarize it for you.

02:29 - 02:32: The text can be summarized into the following distinct topics:
1. Uploading pictures
2. Personal choice and control

02:32 - 02:35: 1. Current state of the project: The text indicates that the project is being kept in its current state.
2. Action being taken: The individual plans to click "Start Building" to initiate a process or task.

02:35 - 02:37: The text "Now just continuing" is too brief to be summarized into distinct topics.

02:37 - 02:39: 1. Introduction of agents
2. Initiation of work by agents

02:39 - 02:44: The text can be summarized into the following distinct topics:

1. Proposal for building something
2. List of steps to follow

02:44 - 02:48: 1. User authentication implementation
2. Personalized news feed development

02:48 - 02:50: Topic: Search functionality

02:50 - 02:53: 1. API Endpoint Creation:
- The text mentions the creation of an API endpoint, which involves defining a URL where clients can access specific data or functionality.

2. Caching Implementation:
- The text also refers to implementing caching, which involves storing frequently accessed data in a temporary storage to improve performance and reduce the need to fetch data repeatedly.

02:53 - 02:55: The text "Going to uproot plan" is too short to provide a meaningful summary with distinct topics. Could you provide more context or additional information for me to work with?

02:55 - 02:57: Topic 1: Monitoring progress
Topic 2: Screen display

02:57 - 02:59: Topic 1: Chat feature
Topic 2: Location on the right hand side

02:59 - 03:03: The text can be summarized into one distinct topic:
- Creation of files on the left hand side

03:03 - 03:05: The text appears to be very brief and lacks specific details to be summarized into distinct topics. It seems to refer to a main model scraper, which could potentially be a tool, software, or device used for scraping or extracting data from a source. Without further context or information, it is challenging to provide a more detailed summary or identify distinct topics related to the main model scraper.

03:05 - 03:07: The text is very brief and lacks context, making it difficult to identify distinct topics. However, based on the words provided, it seems to be referring to the creation or organization of templates in a static folder. The potential topics could include:
1. Templates
2. Static folder
3. Organization or management of files

03:07 - 03:08: The text "This is super cool" does not contain enough information to be summarized into distinct topics.

03:08 - 03:12: The text can be summarized into the following distinct topics:
1. Personalization
2. Git version control

03:12 - 03:16: 1. Automatic installation process
2. Required packages installation

03:16 - 03:17: The text is very brief and does not contain much information to summarize into distinct topics.

03:17 - 03:20: The text is too short to provide a meaningful summary with distinct topics. Can you provide more context or additional information for me to work with?

03:20 - 03:22: The text is discussing the various packages that are being attempted to install. The distinct topic would be the different types of packages and their installation process.

03:22 - 03:24: The text can be summarized into one distinct topic: Flask application running.

03:24 - 03:26: 1. API
2. Endpoint

03:26 - 03:32: Topics:
1. Versatility of the system in running both back end and front end
2. Specific features or functions that are liked by the speaker

03:32 - 03:33: I'm sorry, but it seems like there is no text provided for me to summarize into distinct topics. Could you please provide the text so that I can help you with summarizing it?

03:33 - 03:35: The text can be summarized into the following distinct topics:
1. Hacker news
2. Viewer

03:35 - 03:40: Topic 1: Automation - The text mentions the process of taking a screenshot automatically.
Topic 2: Status Monitoring - The purpose of taking the screenshot is to understand the current status of something.

03:40 - 03:42: The text is too short to be summarized into distinct topics.

03:42 - 03:45: The text can be summarized into one distinct topic: coding proficiency.

03:45 - 03:47: The text is too brief to be summarized into distinct topics.

03:47 - 03:50: The text is too short to be summarized into distinct topics.

03:50 - 03:53: 1. Database Integration:
- SQL Alchemy as a tool for integrating with databases.
- Importance of database integration for applications.
- Benefits of using SQL Alchemy for database integration.
- Methods and techniques for integrating with databases using SQL Alchemy.

2. SQL Alchemy:
- Overview of SQL Alchemy as an Object Relational Mapper (ORM).
- Features and capabilities of SQL Alchemy.
- How SQL Alchemy simplifies database interactions.
- Use cases and examples of SQL Alchemy in database integration.

03:53 - 03:58: 1. Flask framework
2. Integration
3. Main file
4. Endpoint

03:58 - 03:59: I'm sorry, but the text "Here we go" is too brief to be summarized into distinct topics. If you provide more information or context, I'd be happy to help summarize it for you.

03:59 - 04:00: The text is too short to be summarized into distinct topics.

04:00 - 04:05: 1. Accessing the console: The text mentions a method for accessing the console by clicking on a plus icon and selecting the console option.

04:05 - 04:10: The text can be summarized into the following distinct topics:
1. Accessing information through a new tab
2. Navigating by clicking on specific elements

04:10 - 04:12: The text is too short to be summarized into distinct topics.

04:12 - 04:13: The text "This is super cool" is too short to be summarized into distinct topics.

04:13 - 04:16: 1. Web scraping: The process of extracting data from websites
2. Data display: Presenting the scraped information in a specific format

04:16 - 04:18: The text "Quick prototype" is very brief and does not provide much information to summarize into distinct topics. However, based on the term "prototype," we can infer that the text may be referring to the creation of a preliminary model or sample of a product or system for testing or demonstration purposes. The distinct topics that could be related to this text include prototyping methods, importance of prototyping in product development, benefits of quick prototyping, and examples of successful prototypes.

04:18 - 04:20: The text is too short to be summarized into distinct topics.

04:20 - 04:25: 1. Time reference: The text mentions the specific time frame of "five minutes ago" and the present moment of "now."
2. Page preparation: The text indicates that all pages are now ready, suggesting a completion or readiness of some sort of document or material.

04:25 - 04:26: The text "Let's see the database" is too brief to be summarized into distinct topics. Can you provide more context or details so I can create a meaningful summary for you?

04:26 - 04:29: The text "So going to Postgresql" does not provide enough information to summarize into distinct topics.

04:29 - 04:32: The text is very brief and does not provide much context for specific topics. However, based on the term "schema," the text could potentially be summarized into the topic of data organization or database structure. The schema may refer to the structure or design of a database, outlining the relationships between different data elements or entities.

04:32 - 04:38: The text can be summarized into the following distinct topics:

```
1. Accessing news items
2. Saving news items in a database

04:38 - 04:44: The text can be summarized into the following distinct topics:
1. Preference for saving data in a database
2. Personal liking for the ability to save data

04:44 - 04:47: The text can be summarized into the following distinct topics:

1. Back end development
2. Front end control

04:47 - 04:50: 1. Efficiency improvement: The text suggests that the development
process will be accelerated, indicating an increase in efficiency.
2. Development process: The focus is on the development process, implying that
there are steps or stages being undertaken to achieve a particular goal.

04:50 - 04:53: The text does not provide enough information to be summarized
into distinct topics.

04:53 - 04:59: The text can be summarized into the following topics:

1. Web browser usage for development
2. Rapid development process
3. Efficiency in creating things online

04:59 - 05:01: The text is very brief and does not provide much information to
summarize into distinct topics.

05:01 - 05:03: The text does not contain enough information to be summarized
into distinct topics.

05:03 - 05:06: The text can be summarized into two distinct topics:
1. Creation of more videos similar to the existing one
2. Encouragement for the audience to stay tuned for future content

05:06 - 05:07: The text is very short and simple, but it can be summarized into
two distinct topics: 1) expressing hope, and 2) referring to a video.

05:07 - 05:09: The text can be summarized into the following distinct topics:
1. Call to action: Encouraging viewers to like, share, and subscribe
2. Appreciation: Thanking viewers for watching
```

1.0.8 Conclusion

In this tutorial, we learned how to automatically generate timestamps for videos using OpenAI's GPT model and Whisper for transcription. The entire process involves: 1. Extracting audio from

the video. 2. Transcribing the audio using Whisper. 3. Summarizing the transcribed text using OpenAI GPT. 4. Associating the summaries with timestamps and formatting them into a readable format.

This solution can be further customized, including integrating different summarization strategies, more sophisticated topic detection, or handling longer videos with finer granularity in timestamp creation.

By following this guide, you'll have a fully automated way to generate meaningful timestamps for video content!