Chat-GPT-based Video Summarization for Smart MOOCs Development Track

Amrutha Ukkalam
ukkalam2(MCS)
ukkalam2@illinois.edu

Aruna Parameswaran
aruna2(MCS)
aruna2@illinois.edu

Nikitha Reddy Sankepally
nikitha6(MCS)
nikitha6@illinois.edu

University of Illinois, Urbana Champaign
Champaign, USA

ABSTRACT

The aim of this project is to enhance the user experience on the SMART MOOCs website by providing concise summaries with timestamps for 1-minute segments of video content. This feature allows users to easily navigate to specific sections of interest within the videos. Additionally, the project implements topic modeling to generate a list of topics covered in each video. By offering these summaries and topic lists, users can quickly find and access the information they are interested in, making their learning experience more efficient and effective.The summary view with timestamps allows users to navigate through videos with ease, accessing specific sections of interest without the need to watch the entire video. The generated list of topics covered in each video gives users an overview of the video's content, making it easier for them to identify videos relevant to their interests. These features empower the users to make the most of their learning journey on the SMART MOOCs platform.

1. MOTIVATION

One common challenge faced by students in Massive Open Online Courses (MOOCs) is the overwhelming amount of video content they need to cover. MOOCs often provide video lectures and presentations as a primary means of delivering course material. However, these videos can be lengthy and time-consuming to watch, leading to difficulties in information retention and engagement. Artificial Intelligence (AI) can indeed play a significant role in addressing this problem. The proposed system aims to use AI to generate 1-minute segment summaries of MOOC videos to benefit both students and educators. The proposed system summarizes the video content, extracts topics for each video lecture and provides students with a topic and summary view. By providing concise summaries and keywords, students can quickly grasp the crucial information without having to watch the entire video.

2. INTENDED USERS

The intended users for the proposed system that uses Chat GPT to generate 1-minute segment summaries and extract keywords of MOOC videos are:

Students: The system aims to benefit students who are enrolled in MOOCs and face challenges in covering a large amount of video content. By providing concise summaries and topics for each video lecture, the system allows students to quickly understand the essential information and track that point in the video without having to watch the entire video. This can help improve information retention, save time, and enhance overall engagement with the course material.

Educators: The system can also be beneficial for educators who design and deliver MOOC courses. By automatically generating video summaries and extracting topics, educators can ensure that the core concepts and main points of their lectures are effectively conveyed to students. It can also be useful for other individuals interested in accessing summarized versions of MOOC video content, such as professionals seeking to upskill, or individuals looking to explore specific topics in a time-efficient manner.

3. MAJOR FUNCTIONS

3.1 Summary View for Different Video Segments: We display a summary for each 1 min video segment within a MOOC video in a summary view. Instead of watching the entire video, students can access these short, digestible summaries that capture the key points of each segment. This allows for a more efficient consumption of the content while still gaining an understanding of the important information.

3.2 Summarization of Video Segments with Timestamps: We generate concise phrases or sentences that capture the essential information. Additionally, timestamps alongside these summaries are provided, indicating where in the video each segment is located and can reach that point by clicking on the timestamp.

This allows students to quickly navigate to specific parts of the video that are most relevant to their learning needs.

3.3 List of Topics for a Video: We used ChatGPT to extract and present a list of topics covered in each video presented as keywords under each video. This feature helps students get an overview of the subjects addressed in the video lecture. By having a clear outline of the topics, students can understand the structure of the video and identify the specific areas they need to focus on.

4. IMPLEMENTATION DETAILS

4.1 Prerequisite for Implementation

Before proceeding with the implementation phase of the project, several prerequisites needed to be fulfilled. These prerequisites are outlined below:

SmartMOOCs Website: A MOOC (Massive Open Online Course) website already exists, serving as the platform for hosting and delivering educational videos related to Text Retrieval and Text Analysis. The website is functional and contains a collection of video lectures.
MOOC Video Transcripts: Access to the MOOC video transcripts in TXT (text) format was obtained. These transcripts provide a written version of the spoken content within the video lectures.
Timestamp Files (SRT Format): In addition to the video transcripts, access to timestamp files in SRT (SubRip Subtitle) format was acquired. These timestamp files associate specific time intervals within the video with corresponding segments of the transcript. The SRT files are instrumental in synchronizing the video playback with the corresponding transcript segments, enabling users to navigate through the video content and follow along with the transcript.
We received access to the codebase of the existing MOOC website. This access allowed us to integrate new topics and summary view to the platform. It provides the necessary foundation for implementing the enhancements described in the functions above.

For the frontend, we have integrated our features into the existing SmartMOOCs codebase. It uses React.js, with UI elements from the Semantic UI React library. The build system is Node. Most of the frontend code was written in Typescript, and React reads the data stored in JSON formatted files.

4.2 Segmentation

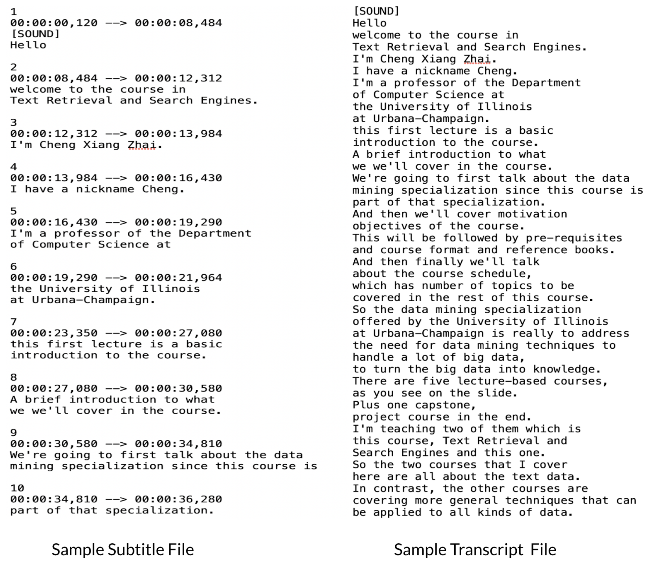
This step involves creating 1-Minute Segments from transcript .txt files and timestamp .srt files. The text data from the transcript files was extracted. The start and end timestamps from the SRT files were retrieved. Using the extracted timestamps, the transcript content was divided into 1-minute segments. Delimiters were added to mark the boundaries of each segment within the transcript. The segmented transcript segments, marked with delimiters, were stored systematically.

By utilizing the .srt timestamp files and the content of the transcript .txt files, the implementation process successfully segmented the transcripts into 1-minute segments. Delimiters were added to clearly identify the boundaries of each segment within the transcript, facilitating further processing and analysis. The video files themselves were not divided into segments; only the corresponding transcript segments were generated.

Procedure followed for the generation of segmented transcripts is as follows:

1. Create an empty dictionary called 'files_dict'
2. For each subtitle file directory path:
   1. Create a tuple of the corresponding text file path and a new file that will contain the segmented and delimited text of the transcript.
   2. Add the tuple to the 'files_dict' dictionary with the srt file directory path as the key.
3. For each key in 'files_dict':
   1. Open the subtitle file, the corresponding text file, the minute segment file.
   2. Get one minute segments from the subtitle file and corresponding text from the text file.
   3. Delimit each segment with '||' and write it to the new file containing the segments.

Figure 1 and Figure 2 provide an initial and final view of the segmentation of tasks. Figure 1 shows a snippet of the subtitle file and the transcript files. Figure 2 shows the newly generated files consisting of the text corresponding to minute segments delimited using the symbol '||'.



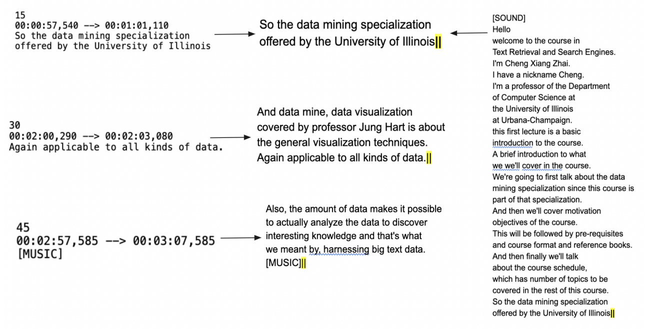
```
1
00:00:00,120 --> 00:00:08,484
[SOUND]
Hello

2
00:00:08,484 --> 00:00:12,312
welcome to the course in
Text Retrieval and Search Engines.

3
00:00:12,312 --> 00:00:13,984
I'm Cheng Xiang Zhai.

4
00:00:13,984 --> 00:00:16,430
I have a nickname Cheng.

5
00:00:16,430 --> 00:00:19,290
I'm a professor of the Department
of Computer Science at

6
00:00:19,290 --> 00:00:21,964
the University of Illinois
at Urbana-Champaign.

7
00:00:23,350 --> 00:00:27,080
this first lecture is a basic
introduction to the course.

8
00:00:27,080 --> 00:00:30,580
A brief introduction to what
we we'll cover in the course.

9
00:00:30,580 --> 00:00:34,810
We're going to first talk about the data
mining specialization since this course is

10
00:00:34,810 --> 00:00:36,280
part of that specialization.
```

Sample Subtitle File

```
[SOUND]
Hello
welcome to the course in
Text Retrieval and Search Engines.
I'm Cheng Xiang Zhai.
I have a nickname Cheng.
I'm a professor of the Department
of Computer Science at
the University of Illinois
at Urbana-Champaign.
this first lecture is a basic
introduction to the course.
A brief introduction to what
we we'll cover in the course.
We're going to first talk about the data
mining specialization since this course is
part of that specialization.
And then we'll cover motivation
objectives of the course.
This will be followed by pre-requisites
and course format and reference books.
And then finally we'll talk
about the course schedule,
which has number of topics to be
covered in the rest of this course.
So the data mining specialization
offered by the University of Illinois
at Urbana-Champaign is really to address
the need for data mining techniques to
handle a lot of big data,
to turn the big data into knowledge.
There are five lecture-based courses,
as you see on the slide.
Plus one capstone,
project course in the end.
I'm teaching two of them which is
this course, Text Retrieval and
Search Engines and this one.
So the two courses that I cover
here are all about the text data.
In contrast, the other courses are
covering more general techniques that can
be applied to all kinds of data.
```

Sample Transcript File

Figure 1: Sample Subtitle and corresponding Transcript file.



```
15
00:00:57,540 --> 00:01:01,110
So the data mining specialization
offered by the University of Illinois
```

So the data mining specialization
offered by the University of Illinois||

[SOUND]
Hello
welcome to the course in
Text Retrieval and Search Engines.
I'm Cheng Xiang Zhai.
I have a nickname Cheng.
I'm a professor of the Department
of Computer Science at
the University of Illinois
at Urbana-Champaign.
this first lecture is a basic
introduction to the course.
A brief introduction to what
we we'll cover in the course.
We're going to first talk about the data
mining specialization since this course is
part of that specialization.
And then we'll cover motivation
objectives of the course.
This will be followed by pre-requisites
and course format and reference books.
And then finally we'll talk
about the course schedule,
which has number of topics to be
covered in the rest of this course.
So the data mining specialization
offered by the University of Illinois||

```
30
00:02:00,290 --> 00:02:03,080
Again applicable to all kinds of data.
```

And data mine, data visualization
covered by professor Jung Hart is about
the general visualization techniques.
Again applicable to all kinds of data.||

```
45
00:02:57,585 --> 00:03:07,585
[MUSIC]
```

Also, the amount of data makes it possible
to actually analyze the data to discover
interesting knowledge and that's what
we meant by, harnessing big text data.
[MUSIC]||

Figure 2. Example of delimited transcript minute segments

4.3 Summarizing 1-Minute Transcript Segments using ChatGPT API

After obtaining the segmented transcript segments for each video in the MOOC website, the next step is to utilize the ChatGPT API to generate summaries for these 1-minute segments. This will enhance the accessibility and usability of the platform by providing concise information about the video content.

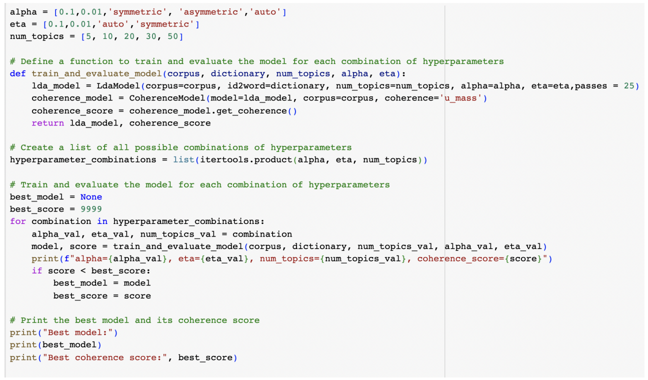
Steps involved::

- Integration with ChatGPT 3.5 API: Establish a connection with the ChatGPT API service to enable communication with the language model.
- Preparing Input for Summarization: Take each 1-minute transcript segment and pass it as input to the ChatGPT API for summarization. The segment can be provided as a text prompt to generate a summary.
- Requesting Summaries: Send requests to the ChatGPT API, passing the text prompts for each 1-minute segment, and receive the corresponding summary generated by the model.
- Storing Summaries: Capture the generated summaries and associate them with their respective 1-minute transcript segments. Store this information with a timestamp in a json file, for easy retrieval and display on the MOOC website.
- Optimization: Free version of ChatGPT has a rate limit of 3 requests per minute, however using the paid version of the website, a maximum of 6000 requests per minute were made. To handle the case where the program could exceed that limit, the API would return a *rate limit error* and to handle this there were minute timeouts introduced and the requests were retried.
- The average length of the one minute segments across the 4 MOOCs provided were ~ 120 words. We attempted at first obtaining a summary in half the number of words (60) and then obtaining the summary in a maximum of 15 words in a maximum of 3 attempts, so we don't exhaust the resources unnecessarily.

By leveraging the ChatGPT API, the implementation will generate concise summaries for each 1-minute transcript segment. These summaries will provide users with a quick overview of the video content, enabling them to navigate through the material more efficiently and effectively.

4.4 Exploration of Topic Modeling using LDA and ChatGPT

In the pursuit of enhancing the MOOC website's features, we considered the addition of keyword topics for each video. To achieve this, we explored the LDA (Latent Dirichlet Allocation) algorithm. Latent Dirichlet Allocation is an unsupervised technique that models topics as word distributions and documents as collections of the word topics. This technique helps identify the main topics discussed in the video lectures. While LDA is commonly used for topic modeling, its effectiveness may vary depending on the dataset and specific use case. In this instance, we observed that the LDA algorithm did not produce the desired outcomes in comparison to the summarization capabilities of ChatGPT. This was due to having less training data. The model was trained on 45 documents (transcripts). At a high level, over the entire MOOC, the model was able to model 5 major topics which were coherent with the 6 topics present in the textbook corresponding to the MOOC, however the performance of the model was poor in topic modeling at the document level.



```python
alpha = [0.1,0.01,'symmetric', 'asymmetric','auto']
eta = [0.1,0.01,'auto','symmetric']
num_topics = [5, 10, 20, 30, 50]

# Define a function to train and evaluate the model for each combination of hyperparameters
def train_and_evaluate_model(corpus, dictionary, num_topics, alpha, eta):
    lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=num_topics, alpha=alpha, eta=eta,passes = 25)
    coherence_model = CoherenceModel(model=lda_model, corpus=corpus, coherence='u_mass')
    coherence_score = coherence_model.get_coherence()
    return lda_model, coherence_score

# Create a list of all possible combinations of hyperparameters
hyperparameter_combinations = list(itertools.product(alpha, eta, num_topics))

# Train and evaluate the model for each combination of hyperparameters
best_model = None
best_score = 9999
for combination in hyperparameter_combinations:
    alpha_val, eta_val, num_topics_val = combination
    model, score = train_and_evaluate_model(corpus, dictionary, num_topics_val, alpha_val, eta_val)
    print(f"alpha={alpha_val}, eta={eta_val}, num_topics={num_topics_val}, coherence_score={score}")
    if score < best_score:
        best_model = model
        best_score = score

# Print the best model and its coherence score
print("Best model:")
print(best_model)
print("Best coherence score:", best_score)
```

Figure 3: Code snippet for hyperparameter tuning of the LDA model.

To handle the challenge of less training data, the model was tested with different values of hyper-parameters. The metric used to test the model performance was the 'u_mass coherence score'. The coherence score is a measure of the semantic similarity between the highest probability words within a single topic. The UMass metric is the co-occurence of highest ranking words in the corpus. This is an intrinsic measure that measures the quality of topics being identified. The intuition of the metric is that if two words are ranked highly in a topic, then it is likely for them to occur together. Since the model is unsupervised, the priors values were unknown and had to be experimented with as shown in Figure 3. The best coherence score the model was able to achieve was -0.907 for 5 topics, with uniform topic prior and word prior of

0.01. LDA presented another challenge of modeling the words to be human interpretable form of topics.
Hence, ChatGPT was chosen for the purpose. Since ChatGPT cannot be fine-tuned to appropriately model topics for very specific documents reliably, the task of topic modeling was converted to a much simpler task of keyword extraction.

4.5 Integrating with UI

The next step in the implementation process was to integrate the summary and topic views within the existing SMARTMOOCs website. After generating the video summaries and topics, the next step in the implementation process was to integrate these features into the existing SMARTMOOCs website. React and Next.js tech stack was used in SmartMOOCs. We set up the development environment for React and Next.js. We implemented React components to display the video summaries and topics. These components will be responsible for fetching the relevant data from json files and rendering it within the user interface.

This allows users to access and interact with the generated summaries and topics, providing them with valuable information and enhancing their learning experience within the platform. These views were designed to be easily accessible and visually integrated into the course page, allowing students to quickly access the summarized content.

To perform the segmentation of the transcript and make API calls to ChatGPT for summarization, Python scripting was utilized. The execution of the script was carried out in Google Colab, a cloud-based Jupyter notebook environment that provides a convenient platform for running Python code and accessing resources like GPUs for accelerated processing.

4.6 Code Workflow

Sections 4.2 and 4.3 provide a detailed description of the algorithms used for summarization and topic modeling. Figure 4 provides an overview of the steps involved in working of the features. The link also provides a detailed overview of the usage and functioning of the features on the actual website.

The following figures provide an overview of high-level code flow as well as the in depth implementation of these modules in Python. Figure 4 provides a high Level overview of code workflow with the main components. Figure 5 provides the Python code for the transcript segmentation and Figure 6 provides the implementation of summarization. The topic retrieval from transcripts was done in a manner similar to summarizing using the ChatGPT API.

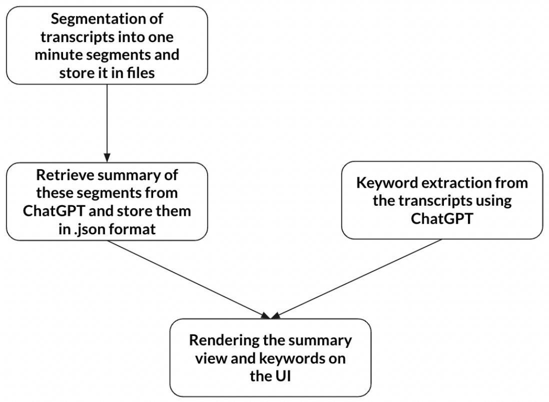
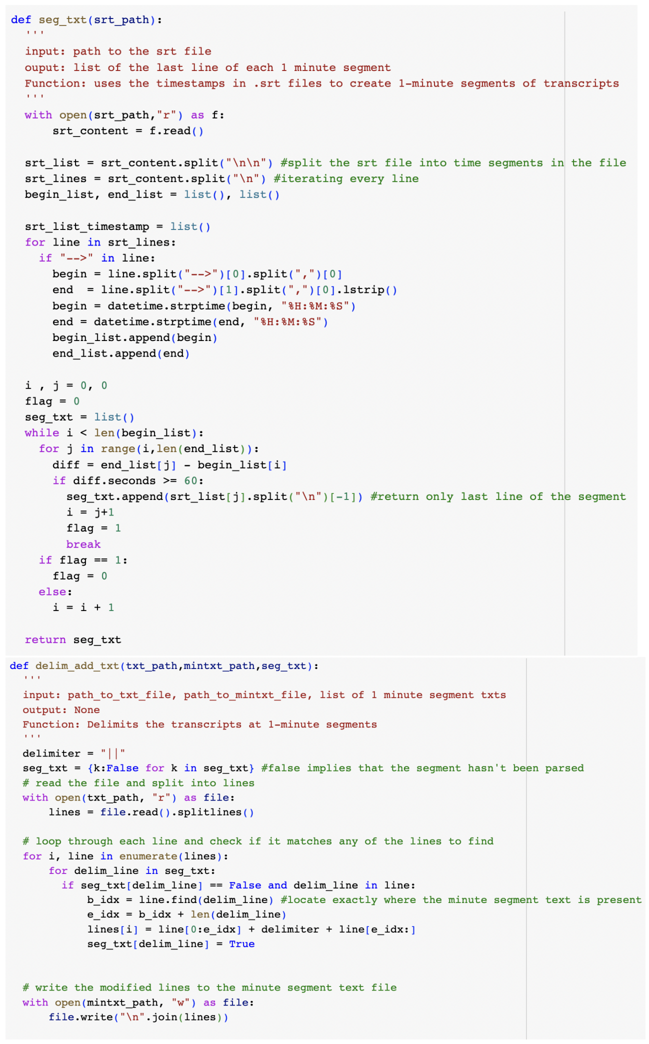
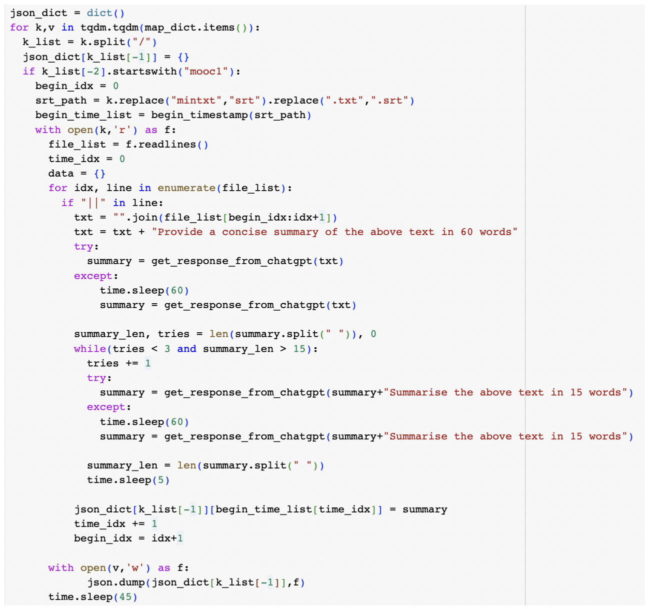

Segmentation of transcripts into one minute segments and store it in files
Retrieve summary of these segments from ChatGPT and store them in .json format
Keyword extraction from the transcripts using ChatGPT
Rendering the summary view and keywords on the UI

Figure 4: High Level overview of code workflow



```
def seg_txt(srt_path):
  '''
  input: path to the srt file
  ouput: list of the last line of each 1 minute segment
  Function: uses the timestamps in .srt files to create 1-minute segments of transcripts
  '''
  with open(srt_path,"r") as f:
      srt_content = f.read()

  srt_list = srt_content.split("\n\n") #split the srt file into time segments in the file
  srt_lines = srt_content.split("\n") #iterating every line
  begin_list, end_list = list(), list()

  srt_list_timestamp = list()
  for line in srt_lines:
    if "-->" in line:
      begin = line.split("-->")[0].split(",")[0]
      end  = line.split("-->")[1].split(",")[0].lstrip()
      begin = datetime.strptime(begin, "%H:%M:%S")
      end = datetime.strptime(end, "%H:%M:%S")
      begin_list.append(begin)
      end_list.append(end)

  i , j = 0, 0
  flag = 0
  seg_txt = list()
  while i < len(begin_list):
    for j in range(i,len(end_list)):
      diff = end_list[j] - begin_list[i]
      if diff.seconds >= 60:
        seg_txt.append(srt_list[j].split("\n")[-1]) #return only last line of the segment
        i = j+1
        flag = 1
        break
    if flag == 1:
      flag = 0
    else:
      i = i + 1

  return seg_txt
def delim_add_txt(txt_path,mintxt_path,seg_txt):
  '''
  input: path_to_txt_file, path_to_mintxt_file, list of 1 minute segment txts
  output: None
  Function: Delimits the transcripts at 1-minute segments
  '''
  delimiter = "||"
  seg_txt = {k:False for k in seg_txt} #false implies that the segment hasn't been parsed
  # read the file and split into lines
  with open(txt_path, "r") as file:
      lines = file.read().splitlines()

  # loop through each line and check if it matches any of the lines to find
  for i, line in enumerate(lines):
      for delim_line in seg_txt:
        if seg_txt[delim_line] == False and delim_line in line:
            b_idx = line.find(delim_line) #locate exactly where the minute segment text is present
            e_idx = b_idx + len(delim_line)
            lines[i] = line[0:e_idx] + delimiter + line[e_idx:]
            seg_txt[delim_line] = True


  # write the modified lines to the minute segment text file
  with open(mintxt_path, "w") as file:
      file.write("\n".join(lines))
```

Figure 5: Code for transcript segmentation



```
json_dict = dict()
for k,v in tqdm.tqdm(map_dict.items()):
  k_list = k.split("/")
  json_dict[k_list[-1]] = {}
  if k_list[-2].startswith("mooc1"):
    begin_idx = 0
    srt_path = k.replace("mintxt","srt").replace(".txt",".srt")
    begin_time_list = begin_timestamp(srt_path)
    with open(k,'r') as f:
      file_list = f.readlines()
      time_idx = 0
      data = {}
      for idx, line in enumerate(file_list):
        if "||" in line:
          txt = "".join(file_list[begin_idx:idx+1])
          txt = txt + "Provide a concise summary of the above text in 60 words"
          try:
            summary = get_response_from_chatgpt(txt)
          except:
              time.sleep(60)
              summary = get_response_from_chatgpt(txt)

          summary_len, tries = len(summary.split(" ")), 0
          while(tries < 3 and summary_len > 15):
            tries += 1
            try:
              summary = get_response_from_chatgpt(summary+"Summarise the above text in 15 words")
            except:
              time.sleep(60)
              summary = get_response_from_chatgpt(summary+"Summarise the above text in 15 words")

            summary_len = len(summary.split(" "))
            time.sleep(5)

          json_dict[k_list[-1]][begin_time_list[time_idx]] = summary
          time_idx += 1
          begin_idx = idx+1

      with open(v,'w') as f:
            json.dump(json_dict[k_list[-1]],f)
      time.sleep(45)
```

Figure 6: Code for summarization using ChatGPT

Usage:

Build & Deploy

Prerequisites

Install npm, node
$ brew install node

Build

Execute the following steps in sequence
$ cd moocs.illinois.edu/
$ npm install
This creates the node_modules/ folder with all the dependencies.

Deploy

Execute the following command
$ npm start dev

Then, in your browser open up http://localhost:3000/. The smart mooc website will be rendered.

5. EVALUATION

5.1 Evaluation of Summarization using ChatGPT:

The summarization process was evaluated multidimensionally. The first being the time required for the successful completion of the process using ChatGPT. The segmentation process for 181 files involved, finding the minute timestamps and corresponding text and writing them to the newly created files took less than 2 mins. However, summarization using ChatGPT without running into rate-limit errors, for 1 MOOC (text-retrieval) that consisted of 45 files of around 10KB, averaging around 13 segments per transcript, each segment having ~120 words, with a time-out intervals of 45 seconds between each file, 5 seconds between each segment and 60 seconds incase of rate-limit reached, the entire process took around 3 hours.

The summaries themselves were evaluated by humans.The summaries for almost all the segments were found coherent and covering all the important aspects of the segment in the summary. It also should be highlighted that without any data pre-processing and segments sometimes being delimited in the midway of a sentence reducing the context, ChatGPT was able to provide very relevant results. It's important to note that the accuracy of summarization is conditional on the quality of transcripts and segments.

5.2 Evaluation of Topic Extraction using ChatGPT:

To provide keyword topics for each video in the MOOC website, the ChatGPT API was utilized to generate the top 5 topics. This approach was chosen as it yielded more efficient and accurate results compared to the LDA algorithm that yielded a coherence score the model was able to achieve was -0.907 for 5 topics, with uniform topic prior and word prior of 0.01. A sample dataset of MOOC video transcripts were utilized to assess the quality and relevance of the results. Send requests to the ChatGPT API, passing the input text, and specify the desired number of topics to be generated (in this case, 5 topics per video). Receive the responses from the ChatGPT API, which will contain the generated topics. Extract the topic information from the response, which may involve parsing or processing the text data to isolate the identified topics. Associate the generated topics with their respective lectures and store this information in a json file.

Upon evaluating the outputs, we found that the ChatGPT APIs consistently outperformed LDA in terms of keyword extraction and topic modeling. The ChatGPT API responses provided more accurate and relevant keywords for each video segment, capturing the core concepts effectively. The prompts used with ChatGPT APIs enabled a focused and context-aware extraction of keywords.

Based on these findings, we made the decision to utilize the ChatGPT APIs for keyword extraction and topic modeling within the integrated system. This approach yielded superior results, ensuring that the generated keywords and topics were highly informative and aligned with the content of the video lectures.

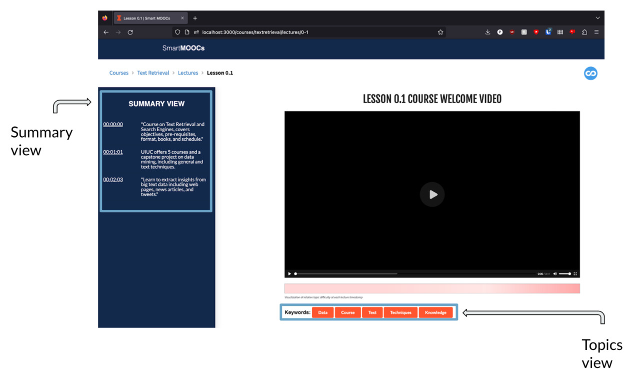
6. CHALLENGES

Quantitative Evaluation of Summaries: Ensuring the accuracy and relevance of the generated summaries can be a challenge. ChatGPT, being a language model, may occasionally produce summaries that are not entirely accurate or may miss important details. Fine-tuning the summarization process and incorporating feedback mechanisms can help address this challenge. Unreliable and inconsistent summaries, i.e, with every request, ChatGPT doesn't provide the same response which hinders a reliable quantitative evaluation.

Rate Limiting and Throttling: ChatGPT API services often have rate limits in place to manage usage and prevent abuse. Exceeding these limits can result in API errors or throttling of requests. Implementing mechanisms to track and manage API rate limits, including appropriate error messaging and rate limit retries, is important to ensure smooth operation. However, when a rate limit error occurred, we implemented a retry mechanism that waits for a certain period of time before making the API request again.

7. RESULTS

Figure 7 provides a design overview of the website highlighting the newly added features: summary view and topics view.


Lesson 0.1 | Smart MOOCs
localhost:3000/courses/textretrieval/lectures/0-1
SmartMOOCs
Courses > Text Retrieval > Lectures > Lesson 0.1
SUMMARY VIEW
LESSON 0.1 COURSE WELCOME VIDEO
Visualization of relative topic difficulty at each lecture timestamp
Keywords: Data Course Text Techniques Knowledge
Summary view
Topics view

00:00:00 "Course on Text Retrieval and Search Engines, covers objectives, pre-requisites, format, books, and schedule."

00:01:01 UIUC offers 5 courses and a capstone project on data mining, including general and text techniques.

00:02:03 "Learn to extract insights from big text data including web pages, news articles, and tweets."

Figure 7. An overview of the UI design

In the examples that follow, summary of minute segments and keywords of the course will be enlisted.

1. Natural Language Content Analysis
Segment: [SOUND] This lecture is about natural language content analysis. As you see from this picture, this is really the first step to process any text data. Text data are in natural languages. So, computers have to understand natural languages to some extent in order to make use of the data, so that's the topic of this lecture. We're going to cover three things. First, what is natural language processing, which is a main technique for processing natural language to obtain understanding? The second is the State of the Art in NLP,which stands for natural language processing. Finally, we're going to cover the relation between natural language processing and text retrieval. First, what is NLP? Well, the best way to explain it is to think about,

Summary: Intro to content analysis for processing text data, related to NLP and text retrieval.

Topics: ["Natural language processing", "State of the art in NLP", "Text retrieval", "Ambiguity", "Bag of words representation"]

2. Vector Space Model- Improved Instantiation
Segment: So suppose, we change the vector to term frequency vectors. Now, let's look at these three documents again. The query vector is the same because all these words occurred exactly once in the query. So the vector is still 0 1 vector. And in fact,
d2 is also essential in representing the same way because none of these words has been repeated many times. As a result, the score is also the same, still three. The same issue for d3 and we still have a 3. But d4 would be different, because now, presidential occurred twice here. So the end in the four presidential in the [INAUDIBLE] would be 2 instead of 1. As a result, now the score for d4 is higher. It's a four now. So this means, by using term frequency,

Summary: Changing to term frequency vectors affects scores in search engines. Increasing frequency increases score.

Topics: ["Vector", "Model", "TF-IDF", "Weighting", "Scoring"]

3. Link Analysis
Segment 1: any spammer to just manipulate the one
signal to improve the ranking of a page. So as a result
people have made a number of major extensions
to the ranking algorithms. One line is to exploit links to
improve scoring and that's the main topic of this lecture.
People have also proposed algorithms to exploit large scale
implicit feedback information in the form of clickthroughs.
That's of course in the category of feedback techniques, and
machinery is often used there. In general, in web search
the ranking algorithms are based on machinery algorithms to
combine all kinds of features. And many of them are based
on the standard original models such as BM25 that we talked about, or queried iCode to score different parts of documents or to, provide additional features based on content matching.

Segment 2: But link information is also very useful so
they provide additional scoring signals. So let's look at links in
more detail on the web. So this is a snapshot of some part of the web, let's say. So we can see there are many links that link different pages together. And in this case you can also look at the, the center here. There is a description of a link that's pointing to the document on the right side. Now this description text is called anchor text. If you think about this text, it's actually quite useful because it provides some extra description of that page being pointed to. So, for example, if someone wants to bookmark Amazon.com front page, the person might say, the big online

bookstore, and then with a link to Amazon, right?

Summary: Multiple signals in ranking algorithms prevent spamming. Features include links, clickthroughs, and queried iCode.

Links and anchor text provide scoring signals for search engines and give extra page descriptions.

Topics: ["Link analysis", "Web search", "Ranking algorithms", "Information quality", "Anchor text"]

The above examples demonstrate the performance of summarization and topic extraction using ChatGPT. Whether a segment is complete in itself like in example 1 or contains random terms such as d2 and d3 as in example 2 or is incomplete as in example 3, ChatGPT produces relevant summaries.

8. CONCLUSION AND FUTURE WORK

In conclusion, the proposed features aim to enhance the learning experience of students on the SMART MOOCs platform by providing concise summaries with timestamps and a topic list corresponding to each video. The features can prove to be time-saving for students especially when dealing with multiple courses. By leveraging AI techniques such as text summarization and topic modeling, the system offers an efficient and effective way for students to navigate through the overwhelming amount of video content. This project's success can contribute to the growing field of educational technology by enhancing users' experience of using the available resources.

Currently, the website is limited to summary view. A future enhancement could include multiple views such as an examination view or topic view. Segmentation is being done at one minute intervals, this could be enhanced by exploring “smart” segmentation that could potentially segment transcripts based on topics. The aim is also to roll out the features into the SmartMOOC production for students use.

9. MEMBER CONTRIBUTION

Amrutha Ukkalam: Responsible for segmentation, summarization, and exploration of topic-modeling using Latent Dirichlet Allocation .
Aruna Parameswaran: Responsible for topic modeling/key-word extraction, integration of the summary view and keyword view with the SmartMOOC website UI.

Nikitha Reddy Sankepally - Exploration of methods for summary retrieval via ChatGPT prompts (single-shot vs interactive), setting up OpenAI ChatGPT APIs.

10. REFERENCES

1. https://platform.openai.com/docs/api-reference?lang=python
2. https://datascience.oneoffcoder.com/lda.html
3. https://radimrehurek.com/gensim/models/coherencemodel.html
4. Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15). Association for Computing Machinery, New York, NY, USA, 399–408. https://doi.org/10.1145/2684822.2685324
5.https://ts2.space/en/how-chatgpt-helps-improve-text-clustering-and-topic-modeling/