Questionnaire and Notes Generator for Video Lectures

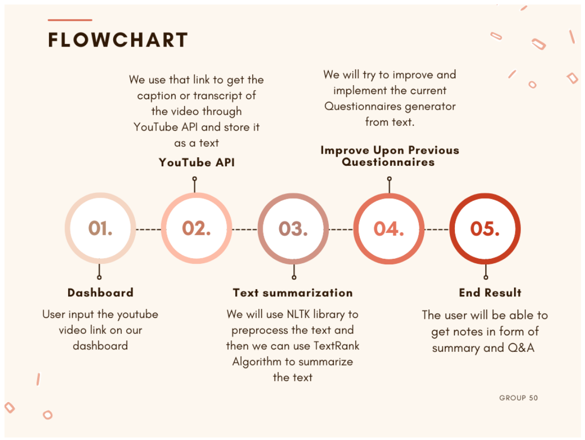
Updated problem statement, our solution, and literature review

Piyush Verma	Mohammad Amaan Khan	Shubham Sahni	Meghna	Prajeet P. Sahoo	Nikita Kapoor
2020097	2020388	2020132	2020080	2020393	2020531

Introduction
With the rise of e-learning, YouTube has become a popular platform for educational content. However, we as students often face difficulties in retaining, recalling, and understanding the information presented in video lectures. The problem arises from the overwhelming amount of information and lack of interactive features. To address this issue, we propose to develop a questionnaire and notes generator for video lectures with the help of artificial intelligence.

Updated Problem Statement
Our tool will help students to develop the text from the videos and summarize the video from youtube they are watching in the form of text. This text summariser will help the students to get a shortened form of the video and get questions based on the summarized text and transcript from the video. Following this, we will be attempting to create a questionnaire generator by identifying a candidate keyword in each of the points in the summarizer. The summarizer produces the output in points which makes it easier for learning, as well as for identification of keywords that will help in generating questions. The questions can be of fill in the blank type, or in the form of e-flashcards. This is our goal to further this project.

Importance of the Problem
Our project could aid several e-learning platforms tremendously. It will increase student engagement as well as the learning prospects of the students. It will help the platforms stay motivated since their viewers are more engaged. Students would not have to go the extra mile to look for materials that better their understanding.


FLOWCHART

01. Dashboard: User input the youtube video link on our dashboard

02. YouTube API: We use that link to get the caption or transcript of the video through YouTube API and store it as a text

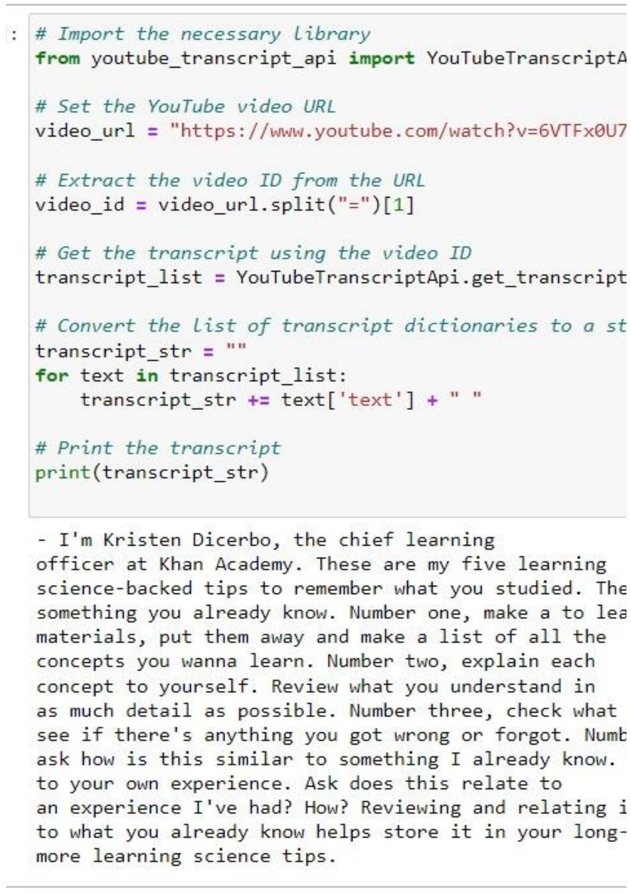
03. Text summarization: We will use NLTK library to preprocess the text and then we can use TextRank Algorithm to summarize the text

04. Improve Upon Previous Questionnaires: We will try to improve and implement the current Questionnaires generator from text.

05. End Result: The user will be able to get notes in form of summary and Q&A

GROUP 50

Scope of the problem
The tool being developed is aimed at school and university students, with the purpose of generating notes and quiz questions for recorded video lectures. It will be web-based and accessible from any device with an internet connection. Using natural language processing and machine learning algorithms, the tool will be able to extract key concepts from the lectures and generate questions accordingly. This tool is expected to give students a convenient and efficient way to study and prepare for their exams. It will also aid the academic YouTube channels and increase their viewership.



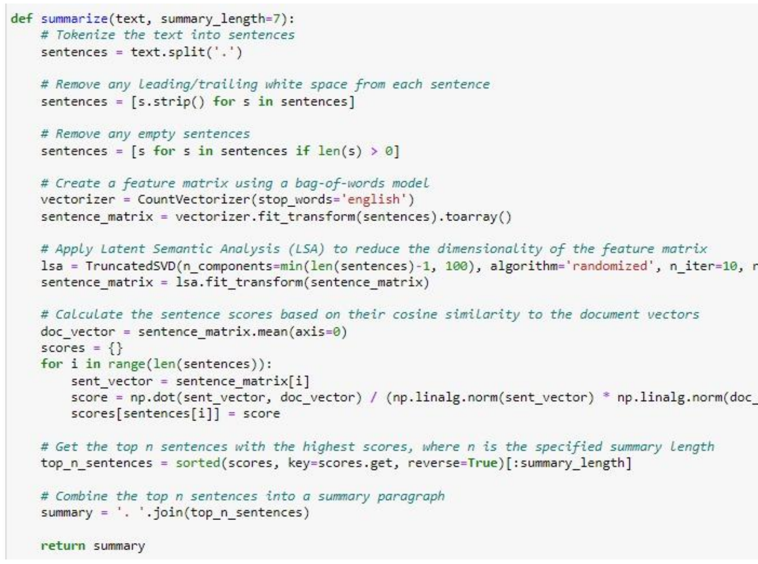
```
# Import the necessary library
from youtube_transcript_api import YouTubeTranscriptA

# Set the YouTube video URL
video_url = "https://www.youtube.com/watch?v=6VTFx0U7

# Extract the video ID from the URL
video_id = video_url.split("=")[1]

# Get the transcript using the video ID
transcript_list = YouTubeTranscriptApi.get_transcript

# Convert the list of transcript dictionaries to a st
transcript_str = ""
for text in transcript_list:
    transcript_str += text['text'] + " "

# Print the transcript
print(transcript_str)
```

```
- I'm Kristen Dicerbo, the chief learning
officer at Khan Academy. These are my five learning
science-backed tips to remember what you studied. The
something you already know. Number one, make a to lea
materials, put them away and make a list of all the
concepts you wanna learn. Number two, explain each
concept to yourself. Review what you understand in
as much detail as possible. Number three, check what
see if there's anything you got wrong or forgot. Numb
ask how is this similar to something I already know.
to your own experience. Ask does this relate to
an experience I've had? How? Reviewing and relating i
to what you already know helps store it in your long-
more learning science tips.
```

Image 1: The above image represents the video-to-text conversion using YouTube API. The video used is from the channel "Khan Academy" and can be found here bit.ly/429MjD2

Updated Baseline Results

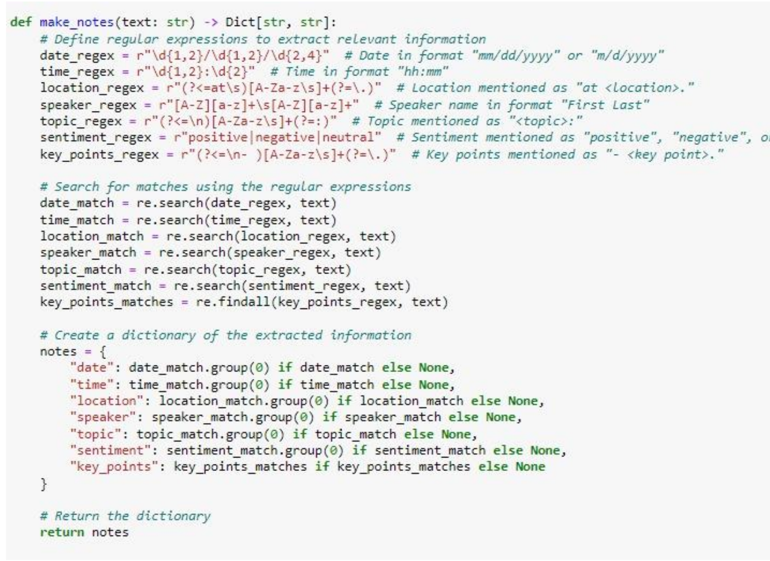
Our project has been fully prepared up to the Summarizer, and we are currently working on some of the algorithms for the questionnaire generator.
The Summarizer has been made using the CountVectorizer function from the scikit-learn library to create a bag-of-words model and Latent Semantic Analysis (LSA) to reduce the dimensionality of the feature matrix.



```
def summarize(text, summary_length=7):
    # Tokenize the text into sentences
    sentences = text.split('.')

    # Remove any leading/trailing white space from each sentence
    sentences = [s.strip() for s in sentences]

    # Remove any empty sentences
    sentences = [s for s in sentences if len(s) > 0]

    # Create a feature matrix using a bag-of-words model
    vectorizer = CountVectorizer(stop_words='english')
    sentence_matrix = vectorizer.fit_transform(sentences).toarray()

    # Apply Latent Semantic Analysis (LSA) to reduce the dimensionality of the feature matrix
    lsa = TruncatedSVD(n_components=min(len(sentences)-1, 100), algorithm='randomized', n_iter=10, 
    sentence_matrix = lsa.fit_transform(sentence_matrix)

    # Calculate the sentence scores based on their cosine similarity to the document vectors
    doc_vector = sentence_matrix.mean(axis=0)
    scores = {}
    for i in range(len(sentences)):
        sent_vector = sentence_matrix[i]
        score = np.dot(sent_vector, doc_vector) / (np.linalg.norm(sent_vector) * np.linalg.norm(doc_
        scores[sentences[i]] = score

    # Get the top n sentences with the highest scores, where n is the specified summary length
    top_n_sentences = sorted(scores, key=scores.get, reverse=True)[:summary_length]

    # Combine the top n sentences into a summary paragraph
    summary = '. '.join(top_n_sentences)

    return summary
```

Image 2

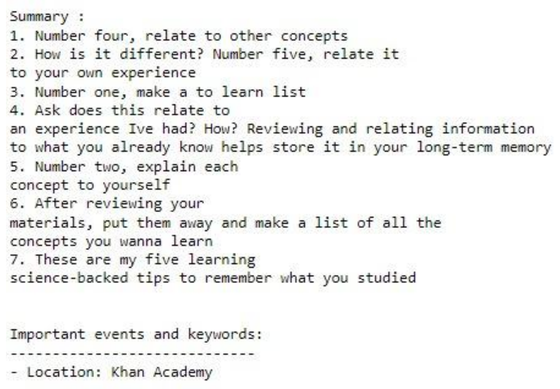
Further, after the dimensionality has been reduced, the cosine similarity between each of the sentences and the average is found across the sentence vectors. The highest scoring vectors are returned. Information Retrieval concepts that have been taught in class have been successfully implemented.



```
def make_notes(text: str) -> Dict[str, str]:
    # Define regular expressions to extract relevant information
    date_regex = r"\d{1,2}/\d{1,2}/\d{2,4}"  # Date in format "mm/dd/yyyy" or "m/d/yyyy"
    time_regex = r"\d{1,2}:\d{2}"  # Time in format "hh:mm"
    location_regex = r"(?<=at\s)[A-Za-z\s]+(?=\.)"  # Location mentioned as "at <location>."
    speaker_regex = r"[A-Z][a-z]+\s[A-Z][a-z]+"  # Speaker name in format "First Last"
    topic_regex = r"(?<=\n)[A-Za-z\s]+(?=:)"  # Topic mentioned as "<topic>:"
    sentiment_regex = r"positive|negative|neutral"  # Sentiment mentioned as "positive", "negative", o
    key_points_regex = r"(?<=\n- )[A-Za-z\s]+(?=\.)"  # Key points mentioned as "- <key point>."

    # Search for matches using the regular expressions
    date_match = re.search(date_regex, text)
    time_match = re.search(time_regex, text)
    location_match = re.search(location_regex, text)
    speaker_match = re.search(speaker_regex, text)
    topic_match = re.search(topic_regex, text)
    sentiment_match = re.search(sentiment_regex, text)
    key_points_matches = re.findall(key_points_regex, text)

    # Create a dictionary of the extracted information
    notes = {
        "date": date_match.group(0) if date_match else None,
        "time": time_match.group(0) if time_match else None,
        "location": location_match.group(0) if location_match else None,
        "speaker": speaker_match.group(0) if speaker_match else None,
        "topic": topic_match.group(0) if topic_match else None,
        "sentiment": sentiment_match.group(0) if sentiment_match else None,
        "key_points": key_points_matches if key_points_matches else None
    }

    # Return the dictionary
    return notes
```

Image 3

Images 2 and 3 are functions that implement techniques to provide us with a summary, which has been shown in image 4 below.



```
Summary :
1. Number four, relate to other concepts
2. How is it different? Number five, relate it
to your own experience
3. Number one, make a to learn list
4. Ask does this relate to
an experience Ive had? How? Reviewing and relating information
to what you already know helps store it in your long-term memory
5. Number two, explain each
concept to yourself
6. After reviewing your
materials, put them away and make a list of all the
concepts you wanna learn
7. These are my five learning
science-backed tips to remember what you studied


Important events and keywords:
-----------------------------
- Location: Khan Academy
```

Image 4

Updated Literature Review

Text Summarizing Using NLP[1]
In this paper, the methods used to summarize the given text come under Natural Language Processing. They have effectively identified the problem, which is that it is often difficult for us to go through an entire article or document. This aligns with our project, where we aim to reduce the effort it takes to watch a full academic video. The research paper used TextRank Algorithm, which identifies the content units, and identifies the relations that append the content units. The end result is based on a scoring scheme such that the highest-level sentences will shape a synopsis. This is a useful algorithm, and it will also aid us in our project and provide a better understanding of how to move forward.

Improved Code Summarization via a Graph Neural Network [2]
The methods used in this paper are very advanced and high performing. They have successfully summarized a source code using Graph Neural Networks. Their approach has been to embed the source code sequence and the AST node tokens and encode the embedding output following a recurrent layer, and Convolutional Graph Neural Networks for the AST nodes and edges. It will then decode the encoder outputs and predict the next token in the sequence. Although the use cases of this article are different from what our project aims to do, it does give us new insight into how a summarization could be performed using GNN. It provided a very high-accuracy code summarizer. A method involving GNN could be experimented with for a problem that aligns with our project. The scope of an algorithm like this is endless. We will still be continuing with TextRank.

Multi-document Summarization via Deep Learning Techniques: A Survey[3]
This paper surveys various algorithms for multi-document summarization, including traditional methods like TextRank and recent deep learning-based approaches such as Seq2Seq models and Transformer-based models like BERT and GPT.
The authors evaluate their proposed approach on several datasets, including the DUC, TAC, and CNN/Daily Mail datasets. They compare their results to several state-of-the-art methods and show that their approach outperforms previous methods in terms of ROUGE scores. They also conduct ablation studies to analyze the effectiveness of different components of their approach.
Overall, the paper provides a comprehensive overview of multi-document summarization algorithms and presents a promising new approach to the task. Using multiple datasets allows for a thorough evaluation of the proposed method and provides a basis for future research in the field.

Student Perceptions towards the Use of YouTube as an Educational Tool for Learning and Tutorials[4]
This paper reviews the perception of students when it comes to using YouTube as an educational tool. The categories of students used were spread across 6 academic departments, and a five point-Likert scale was implemented. The results from the study showed that how useful technology is perceived does influence their use of the said technology. Use of YouTube as a learning platform showed positive results. In our project, we aim to aid academic YouTube channels to help students learn in a more comprehensive manner.

How to create your own Question-Answering system easily with python[5]
The article provides a step-by-step tutorial on building a Question Answering System using Natural Language Processing techniques. The author used pre-trained language models, including BERT and DistilBERT, to extract relevant information from the text and generate answers to user questions. The system also

uses Named Entity Recognition to identify entities in the text and improve answer quality. Additionally, the author employed the Flask framework to deploy the system as a web application.
In our project, we will use similar NLP techniques to extract information from video lectures and generate relevant questions for a questionnaire. Additionally, we aim to incorporate methods to make it easier to generate meaningful questions. We can also consider deploying the questionnaire generation system as a web application using Flask, similar to the author's approach.

Proposed Methods

We want to create a model that can summarize videos. We will use the YouTube API, for which the code has been supplied but we have not yet linked it, to retrieve the transcripts of the YouTube videos. The resulting transcript will also be subjected to text summarizing using the NLTK package to prepare the text. We shall then employ the TextRank Algorithm. We will advance this project by trying to make improvements to earlier Questionnaire generators to produce our desired outcome. We will try to offer students with a platform that combines practice questions and summaries into a single, seamless experience. Academic videos on YouTube are the dataset we'll use for the same. The dataset we will use for the same is academic videos on YouTube provided by the channels “Khan Academy” and “BYJU’S.” We will also try to improve upon some pre existing questionnaire generators that we will be using.

Conclusion
We have provided an updated problem and solution and provided a more critical analysis of the research papers that were found. We created a concrete flow for the remaining semester which involves the successful creation of a text summarizer, and includes a pre-existing questionnaire generator with our own improvements.

Bibliography

[1] Gogulamudi, Vijay & Yadav, Arvind & Vishnupriya, B. & Lahari, M. & Smriti, J. & Reddy, D.. (2021). Text Summarization Using NLP. 10.3233/APC210179.

[2]Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. 2020. Improved Code Summarization via a Graph Neural Network. In Proceedings of the 28th International Conference on Program Comprehension (ICPC '20). Association for Computing Machinery, New York, NY, USA, 184–195. https://doi.org/10.1145/3387904.3389268

[3] Congbo Ma, Wei Emma Zhang, Mingyu Guo, Hu Wang, and Quan Z. Sheng. 2022. Multi-document Summarization via Deep Learning Techniques: A Survey. ACM Comput. Surv. 55, 5, Article 102 (May 2023), 37 pages. https://doi.org/10.1145/3529754

[4] Maziriri, E. T., Gapa, P., & Chuchu, T. (2020). Student Perceptions Towards the use of YouTube as An Educational Tool for Learning and Tutorials. International Journal of Instruction, 13(2), 119-138. https://doi.org/10.29333/iji.2020.1329a

[5] Andre Macedo Farias, Jul 8(2019) bit.ly/3JbIjsP