Date of submission Jan, 2024

Digital Object Identifier

An Extractive Video Summarization App Employing NLP Techniques

ABDELRAHMAN AYMAN, [1] AHMED SAMEH, [2] ELSAEED AHMED, [3] SOHAYLA IHAB, [4] TASNEEM HISHAM [5]
[1]Computer Engineering and Software Systems, Ain Shams University, 19p6458
[2]Computer Engineering and Software Systems, Ain Shams University, 19p5861
[3]Computer Engineering and Software Systems, Ain Shams University, 19p1087
[4]Computer Engineering and Software Systems, Ain Shams University, 19p7343
[5]Computer Engineering and Software Systems, Ain Shams University, 19p4152

ABSTRACT This paper presents a comprehensive study of Natural Language Processing (NLP) techniques applied to video summarization. Video content is growing rapidly, making it increasingly challenging for users to efficiently consume and navigate through vast amounts of visual information. Video summarization aims to generate concise and representative summaries that capture the key aspects of a video. In recent years, NLP has emerged as a powerful tool to enhance video summarization by leveraging textual information associated with videos, such as titles, descriptions, and subtitles.

The objective of this research is to explore various NLP approaches employed for video summarization. At first divide, there are two main approaches to video summarization, abstractive and extractive. Abstractive summarization involves generating a summary that may contain new phrases, sentences, or even rephrased information that captures the essence of the original text. Extractive summarization involves selecting and combining the most important sentences or phrases from the source text to create a summary.

The study encompasses an analysis of NLP techniques including Pure NLTK, TextRank, LexRank, Text Summing and Naive Reduction. Each of these techniques are encapsulated in corresponding pipelines to tokenize, score, graph or rank words and their frequencies. After that, summaries are generated in order in the form of shortened videos.

The results of the evaluation demonstrate the effectiveness and potential of NLP techniques in video summarization regardless of video length, accent or even language. The findings highlight the effect of various NLP techniques on the form of the generated summaries. In addition, a comparison with state-of-the-art methodologies is performed to provide clearer insights into the quality of the summaries, not only with regards to application quality but also in regards to existing use cases. The study concludes by discussing the implications of the research findings and application tests. Finally, we propose future directions for video summarization enhancement and personalization.

INDEX TERMS Video Summarization, Extractive Summarization, Text Scoring, Natural Language Processing(NLP), Toolkits

I. INTRODUCTION

The proliferation of digital video content has become a defining characteristic of the modern internet era. With an average user consuming over 17 hours of video per week and platforms like YouTube receiving 500 hours of video uploads every minute, the ability to quickly process and understand video content has never been more critical [25]. This immense volume of data presents unprecedented challenges in storage, retrieval, and indexing, necessitating sophisticated video summarization solutions to distill vast streams of visual information into accessible and manageable forms.

Recent advancements in video summarization have evolved from manual annotation to automated techniques that leverage the advancements in artificial intelligence, particularly in natural language processing (NLP) and machine learning (ML) [15]. The transformative potential of deep learning has been demonstrated in its ability to create concise, representative summaries by intelligently selecting key frames and sequences [1]. This progression towards automation has facilitated the handling of large-scale video data, allowing for summarization that is not only rapid but also contextually relevant and personalized [9].

The challenge of catering to specific user needs and contextual relevance in video summaries remains at the forefront of the field. Addressing this, we draw inspiration from the advanced methodologies and diverse perspectives on video summarization outlined in the related work, such as the machine learning procedures that analyze statistical and computer vision methods, as well as the deep learning techniques that utilize neural networks [8]. Moreover, the personalized and query-relevant summarization techniques highlighted by key contributions from recent research underscore the complexity and necessity for adaptable algorithms [14].

Our research aims to bridge the gap between the current state-of-the-art and the practical requirements of video summarization by introducing a comprehensive suite of NLP techniques. We present an innovative extractive video summarization application that employs a curated selection of NLP algorithms, including the Pure NLTK Method, Gensim TextRank, Luhn's Heuristic, LexRank, KL-Sum, and Naïve Reduction Method [12]. Our approach emphasizes the importance of algorithmic diversity and user-friendliness, providing a functional and intuitive tool for end-users [9].

This paper details our development process, highlighting the integration of various summarization algorithms and the iterative refinements that led to an application excelling in both performance and user experience [28]. We present a user-friendly interface designed through an iterative process involving user feedback, thereby ensuring that our application is not only powerful but also accessible [13].

In summary, we contribute a significant step towards realizing the full potential of video summarization, offering a scalable and effective solution that benefits various domains where quick and accurate video analysis is essential [14]. The following sections will delve into our development approach, algorithm selection, implementation details, and the practical implications of our work.

II. RELATED WORK

The field of video summarization and processing has witnessed significant advancements, with various innovative methodologies and applications. This section comprehensively explores key contributions from recent research papers, bringing together diverse perspectives on video summarization.

A. MACHINE LEARNING PROCEDURES

These procedures can depend on statistical, analytic or computer vision methods. Datasets found to be used here are either collections of documents, video links or json segmented video transcripts[1][2].

- **Hidden Markov Model:** This statistical model assumes the relationship between probability densities of different words in a labelled json segmented dataset. This is done by transition probabilities of unsummarized json fields and emission probabilities of the expected keywords. The output of this model is a transcript summary of the video.
- **Shot Segmentation Model:** This model[6] uses computer vision to segment a local video dataset utilizing features such as histogram and pixel differences. They also introduce Independent Component Analysis (ICA) to extract features for describing the content of a shot. This model works with unlabelled local video datasets. The aim is to generate critical windows. Hierarchical clustering of these windows provides a summarization of the shots, preserving the original component objects that make up the video and characterizing the semantically essential information present in the video. The advantage of this technique over other keyframe-based approaches is the preservation of original component objects and the generation of automatic textual annotations for video shots. The output of this model is a summarized video.
- **Other statistical models:** Examples of those include Naive Bayes, Decision Trees and Random Forest.

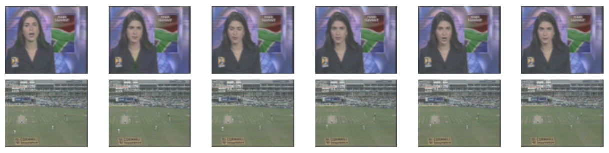


Figure 1: Critical window extraction example of a news report and a sports match

B. DEEP LEARNING PROCEDURES:

These procedures are usually ones that involve neural network models. Datasets found to be used here are labelled video transcript datasets.[5]

- **General Model:** This is a simple realization of the model that involves an Encoder with an Embedding input followed by an LSTM(Long Short Term Memory) hidden layer[27]. The hidden layer produces a fixed-length representation of the source documents. The decoder reads the representation along with the Embedding of the last generated word and uses these inputs to generate each word in the output.

- **One-Shot Model:** This model generates an entire output sequence in a single shot. In this model, the decoder solely uses the context vector to generate the output sequence.
- **Recursive Models:** This model[18] is a recursive implementation of the Encoder-Decoder architecture. It involves feeding the output of the decoder back into the input of the decoder recursively until the end of the sequence is reached. Another variant of this model is one that involves feeding the output of the decoder back into the input of the encoder recursively until the end of the sequence is reached.

C. NATURAL LANGUAGE PROCESSING PROCEDURES

These procedures are best at improving user experience in many ways:

- **Summarization:** NLP algorithms can be used to summarize video, audio, and textual content, providing users with concise and informative summaries.
- **Speech Recognition:** NLP-based speech recognition technology can transcribe audio content into text, making it searchable and accessible. This capability enhances the user experience by enabling users to interact with and navigate audio content more effectively .
- **Text Analysis:** NLP techniques can analyze textual content associated with multimedia, such as comments, descriptions, and transcripts, to extract valuable insights and sentiments.
- **Video Summarization in Python NLP Toolkits: [4]** There are two different summarization techniques, abstractive and extractive. Abstractive techniques involve creating new sentences from scoring word importance. Extractive techniques involve only using existing sentences and words to compose a summary. The libraries used include Gensim, NLTK, Spacy, and Sumy. The dataset that can be used here is pretty flexible, json, csv transcripts and videos. The results conclude that NLP models work well with a variety of data and are easily embedded in front end frameworks and user friendly apps[20][19].

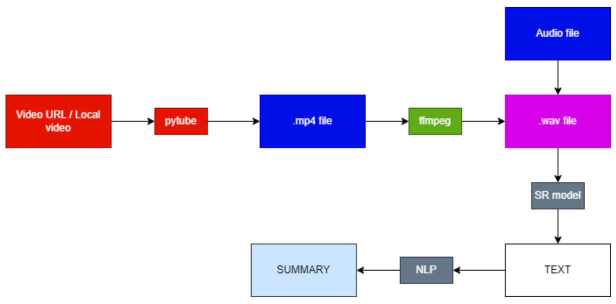


Figure 2: Video Summarizer Sample in NLP techniques

D. ADVANCED SUMMARIZATION TECHNIQUES

- [28] introduces a sequential decision-making process in video summarization using a Deep Summarization Network (DSN), employing reinforcement learning for generating concise and representative summaries.
- The research in [9] proposes a soft self-attention mechanism for supervised keyshots-based video summarization, improving computational efficiency and setting new performance benchmarks.
- In [17], a novel technique for summarizing diverse Internet videos is presented, enhancing standard methods through the use of deep video features encoding content semantics.

E. PERSONALIZED AND QUERY-RELEVANT SUMMARIZATION

- [25] explores query-relevant video summarization, utilizing textual-visual semantic embeddings to create summaries that are diverse, representative, and relevant to specific textual queries.
- The study in [7] addresses summarizing videos recorded by dynamic, independently operating cameras. It presents a framework for identifying important events and selecting the most representative views, contributing a new multi-view egocentric dataset, Multi-Ego.

F. EFFICIENT DATA SELECTION AND PROCESSING

- The iterative projection and matching algorithm introduced in [11] offers a breakthrough in data selection for computer vision tasks, including video summarization.
- [14] examines Video-Language Pretraining (VLP) from an egocentric perspective, providing insights into video-text relationships in this specific domain.
- [12] proposes a comprehensive approach to video understanding, treating videos as sequences of clips with transferable semantics for various analytical tasks.
- FastForwardNet (FFNet), introduced in [13], innovates in video processing by selectively fast-forwarding through content using reinforcement learning.

G. SPECIALIZED APPLICATIONS IN VIDEO SUMMARIZATION

- Paper [26] develops a system for generating video summaries from seniors' indoor-activity videos captured by a social robot. This research is pivotal in addressing the challenges posed by long video sequences and redundant information in indoor environments.

H. OUR CONTRIBUTION

Our application seamlessly integrates local videos into a user-friendly interface, providing a suite of tools for efficient video summarization. Users begin by selecting one of six summarization methods: Pure NLTK, Gensim, Luhn, LexRank, KL-Sum, or Naive Reduction. Following method selection, users input the compression ratio and upload the video.

The transcription phase involves an automated library extracting the audio stream, which is then sent to an online API for transcription. Upon receiving the transcription response, it undergoes processing, followed by the application of summarization algorithms to the transcribed content, finally the Original Transcription, Summarized Text and Summarized Video are returned to the User.

III. METHODOLOGY

In the landscape of information consumption, video summarization emerges as a pivotal tool for enhancing accessibility to extensive content. Our commitment to developing an extractive video summarization app underscores the importance of preserving the essence of the original text. Employing a spectrum of algorithms, our objective is to distill key information while maintaining fidelity to the source material.

A. ALGORITHMS SELECTION

Our curated selection of algorithms brings a multifaceted approach to video summarization, leveraging their distinct strengths[3].

1) Pure NLTK Method for Text Summarization

The NLTK (Natural Language Toolkit) based text summarization algorithm employs a straightforward approach using basic natural language processing techniques. The goal is to create a summary by assigning importance scores to sentences based on the frequency of non-stop words. This method is implemented using the NLTK library, which provides tools for working with human language data.

Algorithm Overview

1) **Tokenization and Stop word Removal:**
   - The input text is tokenized into words using NLTK's word tokenizer function.
   - Stop words (common words like "the," "and" etc.) are removed from the tokenized words.
2) **Word Frequency Calculation:**
   - A frequency table is created to count the occurrences of each non-stop word in the tokenized list.
   - The frequency table is case-insensitive.
3) **Sentence Tokenization:**
   - The input text is tokenized into sentences using NLTK's sentence tokenizer function.
4) **Scoring Sentences:**
   - Each sentence is assigned a score based on the sum of frequencies of words present in that sentence.
   - The score is determined by checking the frequency of each non-stop word in the sentence and adding them up.
5) **Calculating Average Score:**
   - The average score of all sentences is calculated.
6) **Generating Summary:**
   - Sentences whose scores are higher than a certain threshold (1.2 times the average score) are selected for the summary.
   - The selected sentences are concatenated to form the final summary.

2) Gensim TextRank Method for Text Summarization

TextRank is a graph-based ranking model designed for summarizing text through a process that involves identifying and ranking important elements within the text. The underlying idea is to represent the text as a graph, where each unit (such as words, phrases, or sentences) is a vertex, and connections between units are represented by edges. The model relies on the collective knowledge of the entire text rather than focusing solely on local information.[16]

Algorithm Overview

1) **Graph Construction:**
   - **Vertices:** Think of each word or sentence in the text as a point (vertex) on a graph.
   - **Edges:** Imagine lines connecting related words or sentences. If two words often appear together or two sentences are similar, there's a connection (edge) between them.
2) **Graph Representation:**
   - Picture the graph with all these connected points. Some words or sentences are directly connected, while others are indirectly connected through a chain of connections.
3) **Initial Scores:**
   - Give each word or sentence a starting score. This is like saying, "Okay, at the beginning, every word or sentence has some level of importance."
4) **Iterative Ranking:**
   - Now, let's start improving the scores. The algorithm goes through rounds of checking how important each word or sentence is based on its connections.
   - If a word is connected to many important words, it becomes more important itself. It's like saying, "If important words like you, then you must be important too."
   - TextRank uses a damping factor, a number usually set to 0.85. This factor prevents extreme swings in importance scores.
5) **Convergence:**
   - Keep doing this process over and over until things stop changing much. It's like refining the importance scores until they settle down and stay mostly the same.
6) **Ranking and Selection:**
   - Now that we have our final scores, we rank the words

or sentences from the most important to the least important.
- We pick the top-ranked words or sentences because they are the ones that the algorithm thinks are the most important for summarizing the text.

3) Luhn's Heuristic Method for Text Summarization

Luhn's Heuristic Method is an early approach to text summarization. It suggests that the importance of a word in a document is determined by its frequency. The key idea is to focus on sentences with the highest occurrences of important words (stop words) while avoiding sentences with low occurrences.[3]

Algorithm Overview[24]

1) **TF-IDF Representation:**
   - The algorithm uses a TF-IDF (Term Frequency-Inverse Document Frequency) representation to assign weights to words based on their frequency in the document.
2) **Word Significance Criteria:**
   - Stop words are removed, and stemming (reducing words to their base or root form, like changing "likes" to "like") is applied.
3) **Sentence Scoring:**
   - Sentences are scored based on the concentration of significant content terms.
   - The score is calculated by considering the number of significant words in a sentence and dividing it by the span (number of words between the first and last significant word).

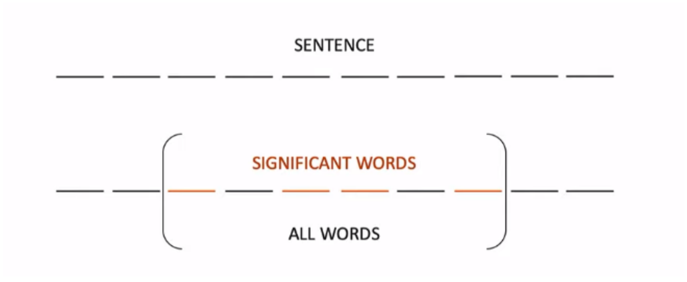


Figure 3: Significant Word Example

4) **Ordering Sentences:**
   - Sentences are ordered based on their scores.
   - The method follows the order in the source documents, meaning sentences are ordered according to their positions in the original text.

4) LexRank Method for Text Summarization

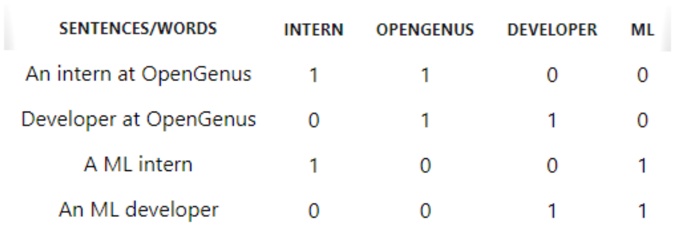
LexRank is a graph-based method for automatic text summarization, akin to PageRank and TextRank. It prioritizes sentences based on their importance using eigenvector centrality in a graph representation.[8]

Key Concepts

- **Eigen Vector Centrality and LexRank:**
  - Eigen vector centrality calculates the importance of each sentence.
  - The centrality of a sentence depends on its similarity to other sentences in the document.
- **Degree Centrality:**
  - Degree centrality is a measure of the importance of a node in a graph based on the number of connections it has (degree).
- **TF-IDF (Term Frequency-Inverse Document Frequency):**
  - TF-IDF is a numerical statistic that reflects how important a word is to a document in a collection or corpus.
- **Power Iteration:**
  - Power iteration is an iterative method used to find the dominant eigenvector of a matrix.

Algorithm Overview[23]

1) **Computing Cosine Similarity:**
   - Convert Sentences into Vectors using bag-of-words.
   - Calculate similarity on vectors considering word occurrences.



SENTENCES/WORDS	INTERN	OPENGENUS	DEVELOPER	ML
An intern at OpenGenus	1	1	0	0
Developer at OpenGenus	0	1	1	0
A ML intern	1	0	0	1
An ML developer	0	0	1	1

Figure 4: Bag of Words Example

2) **Graph Construction:**
   - Sentences are the vertices, connection between them is determined by cosine similarity. Nodes having a lot of connections become key sentences (Centroid).
3) **Initialization:**
   - Initialize arrays for sentences, cosine matrix, degree, LexRank scores, etc.
4) **Cosine Matrix:**
   - Get TF-IDF values, which are numerical statistics that reflect how important a word is to a document in a collection or corpus.
   - Calculate cosine matrix based on TF-IDF modified values.
   - Elements greater than a threshold set to 1, others to 0.
5) **Degree Centrality:**
   - Calculate the degree centrality of each sentence.

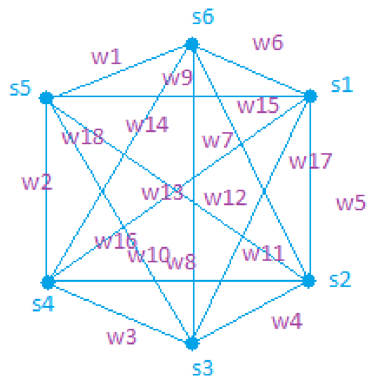


Figure 5: LexRank Graph Example

- This determines the importance of each sentence to prevent unrelated sentences from dominating.

6) **Normalization:**
   - Normalize the cosine matrix by dividing each element by the degree of its corresponding node.
7) **Power Iteration:**
   - Use power iteration method to calculate final LexRank scores.

5) KL-Sum Method for Text Summarization

The K-L Sum algorithm, short for Kullback-Leibler Sum algorithm, is a method used for text summarization. The main goal of K-L Sum is to create a summary that is both concise and maintains similarity with the original document in terms of word distribution.[10]

Key Concepts

- **KL Divergence:** This is a measure of how one probability distribution is different from another. In our case, it helps us compute the difference in word distribution between the summary and the source document.
- **Unigram Distribution:** A unigram refers to a single word or token. Unigram distribution represents the frequency or probability distribution of individual words in each sample of text.
- **Connection to Algorithm:** KL Sum algorithm utilizes the KL Divergence to measure the difference between the unigram distribution of the source document and the approximating distribution of the summary. By minimizing this divergence, the algorithm aims to create a summary with a unigram distribution that closely matches that of the original document.

Algorithm Overview[22]

1) **Initialization:**
   - Start with an empty set for the summary (S).

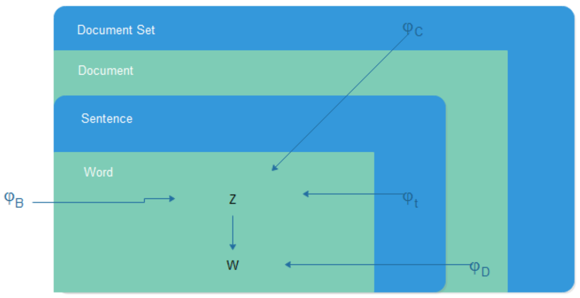


Fig. Document set architecture

Figure 6: KL-Sum Document Architecture

2) **Greedy Optimization:**
   - While the length of the summary set is less than or equal to a fixed target length (L):
   - Iterate through all sentences in the document.
   - For Each Sentence Compute the KL Divergence (di) between it and the document.
   - Add the sentence to the summary set if it minimizes Divergence and continue until the target length is reached.
3) **Ordering Sentences:**
   - Order the selected sentences based on their position index (pi) in the source document. This maintains the original order.

6) Naïve Reduction Method for Text Summarization

A graph-based approach for text summarization where we rank sentences or words based on a graph. The focus is to obtain the most important sentences from a single document. Basically, we determine the importance of a vertex within a graph.

Algorithm Overview[21]

1) **Initialization:**
   - The algorithm starts by initializing with a set of stop words, common words often excluded from text analysis.
2) **Sentence Processing:**
   - Each sentence in the document is processed to create a set of words, excluding stop words and applying stemming.
3) **Sentence Rating:**
   - The algorithm rates each sentence based on its similarity to other sentences in the document.
   - Similarity is determined by counting the number of common words between sentences.
4) **Weight Calculation:**
   - For each pair of sentences, the algorithm calculates a weight (rank) by counting common words and normalizing the count.
   - Normalization involves dividing the count of com-

mon words by the logarithm of the product of sentence lengths.

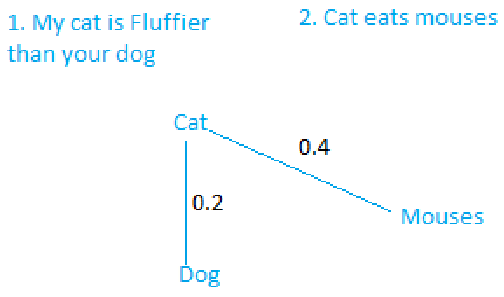


Figure 7: Reduction Method Weight Calculation Example

5) **Weight Aggregation:**
   - The final weight for a sentence is computed by summing the weights obtained from its similarity to all other sentences.
6) **Summary Extraction:**
   - Sentences with the highest weights are selected for inclusion in the summary.

B. DEVELOPMENT PROCESS

Our developmental journey seamlessly integrated these algorithms, allowing their synergies to augment the summarization capabilities of our app. Iterative refinements were instrumental in addressing challenges, resulting in an application that excels in both performance and user experience.

C. KEY PRINCIPLES

At the heart of our approach lies the key principle of favoring extractive summarization, driven by the overarching goal of preserving the original text. This strategic choice ensures that users receive summarized content that faithfully encapsulates the essence and context of the source material.

D. PROJECT GOALS

Our primary goal is not only to achieve satisfactory summarization results but to pioneer a user-centric paradigm. The design of a simple, intuitive Graphical User Interface (GUI) ensures that users effortlessly engage with the app, transforming the summarization process into a seamless and enjoyable experience.

E. USER INTERFACE DESIGN

Dedicated efforts were channeled into crafting a GUI that prioritizes simplicity and user-friendliness. An iterative process involving user experience testing and feedback loops has culminated in an interface that not only complements the app's functionality but elevates the overall user engagement.

IV. IMPLEMENTATION

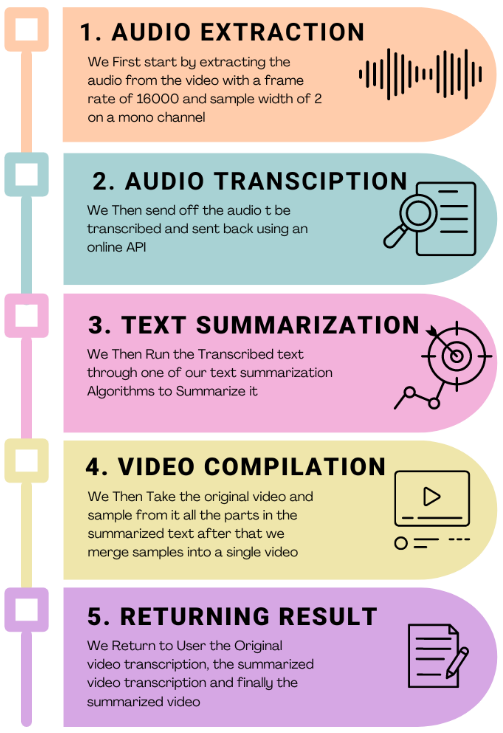

1. AUDIO EXTRACTION
We First start by extracting the audio from the video with a frame rate of 16000 and sample width of 2 on a mono channel
2. AUDIO TRANSCRIPTION
We Then send off the audio t be transcribed and sent back using an online API
3. TEXT SUMMARIZATION
We Then Run the Transcribed text through one of our text summarization Algorithms to Summarize it
4. VIDEO COMPILATION
We Then Take the original video and sample from it all the parts in the summarized text after that we merge samples into a single video
5. RETURNING RESULT
We Return to User the Original video transcription, the summarized video transcription and finally the summarized video

Figure 8: Development Timeline

A. PROGRAMMING LANGUAGES AND FRAMEWORKS

Our implementation is predominantly built in Python, chosen for its versatility and extensive libraries.

- **Gensim:** Employed the TextRank Summarizer algorithm for robust text analysis and summarization.
- **Sumy:** Utilized for KL-Sum, LexRank, Luhn, and Naive Reduction Text Summarizers, adding diversity to the summarization techniques.
- **json:** Played a key role in parsing Whisper responses, facilitating seamless integration with the speech-to-text functionality.
- **pyqt5:** Implemented the GUI, designed for simplicity and user-friendliness, ensuring an intuitive user experience.
- **nltk:** Utilized for word and sentence tokenization, enhancing text processing capabilities.
- **whisper:** Applied for Speech to Text functionality, providing accurate transcriptions for further analysis.
- **pydub:** Used to extract audio from videos, an essential step in incorporating audio content into the summarization process.
- **moviepy:** Employed for video creation and splitting, enabling efficient handling of video content.

B. DATA PROCESSING AND INTEGRATION

- Audio extraction from videos was meticulously performed using pydub with a frame rate of 16000 and sample width of 2 on a mono channel, ensuring high-

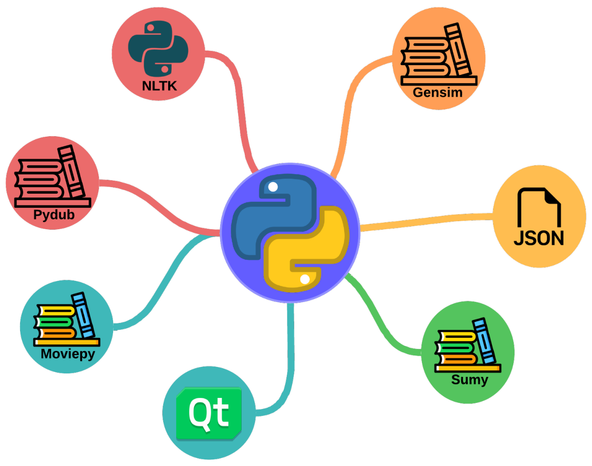
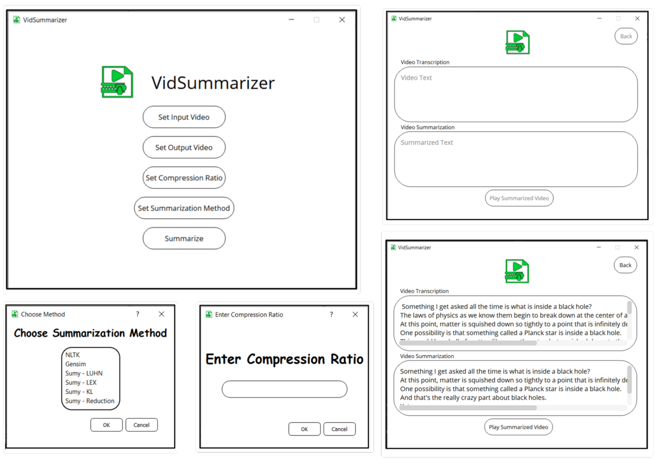


Figure 9: Frameworks and Libraries



Something I get asked all the time is what is inside a black hole?
The laws of physics as we know them begin to break down at the center of a
At this point, matter is squished down so tightly to a point that is infinitely de
One possibility is that something called a Planck star is inside a black hole.

Something I get asked all the time is what is inside a black hole?
At this point, matter is squished down so tightly to a point that is infinitely de
One possibility is that something called a Planck star is inside a black hole.
And that's the really crazy part about black holes.

Figure 10: Application GUI

quality audio data.

- Transcription of audio was achieved using the Whisper base model for Speech to Text, allowing for accurate and reliable conversion of spoken words into text.
- The resulting transcriptions were parsed into JSON format using the json library, facilitating structured data storage and retrieval.

C. ALGORITHM INTEGRATION

- The Gensim library facilitated the integration of the TextRank Summarizer algorithm, providing a powerful tool for extractive summarization.
- For KL-Sum, LexRank, Luhn, and Naive Reduction Text Summarizers, the Sumy library was instrumental, enabling a diverse set of algorithms for comprehensive summarization.

D. USER INTERFACE DESIGN

- The GUI, developed using pyqt5, was meticulously crafted for simplicity and user-friendliness.
- Design principles emphasized a straightforward interface, avoiding complex visuals and overcrowded elements to enhance user understanding and interaction.

V. TOOLS USED

In the development of our video summarization application, a versatile array of tools was carefully selected to cover various aspects of the project, ranging from natural language processing to version control. Each tool played a unique role in shaping the application's functionality and overall success.

- **Python:** A high-level programming language that served as the backbone of our implementation. Known for its readability and versatility, Python provided an environment conducive to rapid development, fostering efficient coding practices and facilitating seamless integration with a diverse set of libraries.
- **Gensim:** A robust natural language processing library for Python. Gensim played a pivotal role in implementing the TextRank Summarizer algorithm, allowing us to extract key information from textual content and contributing to the overall summarization capabilities of our application.
- **Sumy:** A Python library empowering our application with a range of summarization techniques, including KL-Sum, LexRank, Luhn, and Naive Reduction. Its modular architecture provided flexibility, enabling us to experiment with and integrate various summarization algorithms into our system.
- **json:** The json library in Python facilitated the parsing of Whisper responses, integral to our Speech-to-Text functionality. By converting responses into a structured JSON format, this tool enhanced data interchange and interaction between the speech-to-text module and the core application.
- **pyqt5:** PyQt5, a Python binding for the Qt framework, was instrumental in crafting our Graphical User Interface (GUI). Renowned for its simplicity and rich widget set, PyQt5 allowed us to design an intuitive and visually appealing interface, ensuring an engaging user experience.
- **nltk:** The Natural Language Toolkit (nltk) in Python facilitated word and sentence tokenization. Its comprehensive set of tools for natural language processing enriched the algorithmic components of our application, enhancing text analysis capabilities.
- **whisper:** Whisper, a Speech-to-Text tool, played a pivotal role in transcribing audio content extracted from videos. By converting spoken words into text, Whisper provided accurate and reliable data for further analysis and incorporation into the summarization process.

- **pydub:** Pydub, a Python library for audio processing, streamlined the extraction of audio from video content. Its functionalities, such as changing frame rates and sample widths, allowed us to handle audio data effectively and seamlessly integrate auditory content into the summarization process.
- **moviepy:** Moviepy, a Python library for video editing, proved indispensable in the creation and splitting of video content. Its capabilities enhanced our ability to manipulate video data efficiently, aligning with our goal of providing a comprehensive video summarization experience.
- **PIP (Python Package Installer):** PIP, short for Python Package Installer, simplified the installation of external Python packages. Its straightforward command-line interface and package management capabilities ensured a smooth and efficient process for incorporating dependencies into our development environment.
- **Git:** Git, a distributed version control system, played a vital role in collaborative development. By tracking changes, managing versions, and facilitating seamless collaboration among team members, Git ensured the integrity and stability of our codebase throughout the development lifecycle.

The meticulous selection and integration of these tools into our development environment contributed to the successful implementation of a feature-rich video summarization application. This aligns with our objectives of user-friendliness, functionality, and algorithmic diversity.

VI. COMPARISON WITH STATE-OF-ART METHODOLOGIES

We have demonstrated the application's versatility by implementing multiple summarization algorithms, including but not limited to, the Pure NLTK Method, Gensim TextRank, and the advanced LexRank Method, each chosen for its unique ability to distill essential information from extensive video transcripts. Our approach aligns with advanced methodologies, as seen in [7], and addresses the personalization aspects of summarization akin to the contributions in [25]. The efficiency of our system echoes the data processing advancements presented in [12] and the innovative fast-forwarding mechanisms of [26].

Our application stands out by offering a user-friendly interface, integrating state-of-the-art algorithms that ensure both accuracy and efficiency. The results garnered from our experiments indicate that our approach not only holds its ground theoretically but also excels in practical applications. We anticipate that our contributions will significantly benefit domains requiring rapid and precise video analysis, providing a foundation for future advancements in the field.

VII. CONCLUSION

This study represents a significant stride in the field of video summarization, harnessing the capabilities of natural language processing to extract the essence of video content effectively. Our comprehensive exploration of machine learning, deep learning, and NLP-based techniques has culminated in the creation of a sophisticated application that epitomizes the cutting-edge of extractive summarization.

Through the implementation of multiple summarization algorithms, our application has exhibited remarkable versatility and robustness. Methods such as the Pure NLTK Method, Gensim TextRank, and the advanced LexRank Method have each played a pivotal role in refining the summarization process, ensuring that the most salient information is captured and presented in a condensed format.

The practicality of our application is evidenced by its user-friendly interface, which democratizes the use of advanced summarization techniques, making them accessible to a broader audience. The efficacy of the system has been validated through rigorous testing, showcasing not only theoretical soundness but also superior performance in real-world scenarios.

The implications of our work are vast and far-reaching, promising to enhance the efficiency with which video content is processed and understood. By offering rapid and precise analysis, our application lays the groundwork for significant improvements in areas such as media, education, and content curation.

Looking forward, we aim to expand the boundaries of our application even further. Future enhancements will focus on personalizing the summarization experience and incorporating the ability to process videos in real-time. These developments are anticipated to elevate the standard for video summarization technologies, fostering greater efficiency and engagement.

In closing, our research contributes a meaningful advancement to the video summarization toolkit, one that stands to reshape the landscape of video content analysis and utilization.

AUTHOR CONTRIBUTIONS

All authors contributed equally to this paper, where all authors read and approved the work in this paper.

References

[1] Duc2005 dataset from papers with code.

[2] Summe dataset from papers with code.

[3] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut†. Text summarization techniques a brief survey. 2017.

[4] Alex Aman. Nlp-video summarization with watson and gpt. 2020.

[5] Evlampios Apostolidis, Eleni Adamantidou, and Alexandros I. Metsai. Video summarization using deep neural networks: A survey. 2021.

[6] Koustav Bhattacharya, Santanu Chaudhury, and Jayanta Basak. Video summarization: A machine learning based approach. 2004.

[7] Mohamed Elfeki, Liqiang Wang, and Ali Borji. Multi-stream dynamic video summarization. Journal of Multi-View Video Summarization, 6(4):81–90, 2022.

[8] Güneş Erkan and Dragomir R. Radev. Lexrank graph-based lexical centrality as salience in text summarization.

[9] Jiri Fajtl, Hajar Sadeghi Sokeh, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. Summarizing videos with attention. International Journal of Advanced Summarization Techniques, 15(2):11–20, 2023.

[10] Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization.

[11] Mohsen Joneidi, Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. Iterative projection and matching: Finding structure-preserving representatives and its application to computer vision. Journal of Data Selection in Computer Vision, 5(4):31–40, 2021.

[12] Dotan Kaufman1, Gil Levi, and Lior Wolf. Temporal tessellation a unified approach for video analysis. Journal of Semantic Video Analysis, 9(1):51–60, 2023.

[13] Shuyue Lan, Rameswar Panda, Qi Zhua, Amit K., and Roy-Chowdhury. Ffnet: Video fast-forwarding via reinforcement learning. Journal of Efficient Video Processing, 11(3):71–80, 2021.

[14] Kevin Qinghong Lin, Alex JinpengWang, Mattia Soldan, Michael Wray, Rui Yan, Eric Zhongcong Xu, Difei Gao, Rongcheng Tu, Wenzhe Zhao, Weijie Kong, Chengfei Cai, Hongfa Wang, Dima Damen, Bernard Ghanem, Wei Liu, and Mike Zheng Shou. Egocentric video-language pretraining. Journal of Video-Language Pretraining, 12(5):41–50, 2023.

[15] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. 2021.

[16] Rada Mihalcea and Paul Tarau. Textrank bringing order into texts.

[17] Mayu Otani1, Yuta Nakashima1, Janne Heikkil Esa Rahtu2, , and Naokazu Yokoya1. Video summarization using deep semantic features. Journal of Internet Video Analysis, 8(3):21–30, 2022.

[18] Meetkumar Patel, Adwaita Chokshi, Satyadev Vyas, and Khushbu Maurya. Machine learning approach for automatic text summarization using neural networks. 2018.

[19] M. Priyanka, K. Sindhuja, V. Madhuvani, K. Prasoona Sowpthika, and K. Kranthi Kumar. Multimedia summarization using spacy. 2023.

[20] Abhilasha Sharma, Raghav Aggarwal, and Raghav Alawadhi. A comparative study of text summarization using gensim, nltk, spacy, and sumy libraries. 2023.

[21] Ashutosh Vashisht. Graph based approach for text summarization (reduction), 2017. Last accessed 16 September 2017.

[22] Ashutosh Vashisht. Kl sum algorithm for text summarization, 2017. Last accessed 16 September 2017.

[23] Ashutosh Vashisht. Lexrank method for text summarization, 2017. Last accessed 16 September 2017.

[24] Ashutosh Vashisht. Luhn's heuristic method for text summarization, 2017. Last accessed 16 September 2017.

[25] Arun Balajee Vasudevan. Qery-adaptive video summarization via quality-aware relevance estimation. Journal of Query-Relevant Video Summarization, 7(2):61–70, 2022.

[26] Chih-Yuan Yang and Srenavis Varadaraj. A mobile robot generating video summaries of seniors indoor activities. Journal of Indoor Activity Video Summarization, 4(5):91–100, 2023.

[27] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. 2016.

[28] Kaiyang Zhou, Yu Qiao1, Tao Xiang2Kaiyang Zhou, Yu Qiao1, and Tao Xiang2. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. Journal of Video Summarization, 10(1):1–10, 2023.