README2VIDEO

Aditya Chitlangia
achitla@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Firasat Hussain Mohammed
fmohamm8@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Nikhilesh Cherukuri
ncheruk2@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Pankaj Manoharlal Thakur
pmanoha@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Abstract

This project harnesses the power of generative AI to revolutionize how information from GitHub repository README files is communicated, transforming them into immersive video content. Drawing upon the fascinating fact that the human brain processes visual content 60,000 times faster than text, the project aims to exploit this cognitive advantage by combining images, sounds, and movement in its videos. This multimedia approach not only makes the information more organized and navigable but also creates an intuitive, easy-to-understand, fun, and compelling learning experience. By doing so, the project significantly enhances user engagement with GitHub repositories, making complex technical information more accessible and interesting to a wider audience.

Keywords: Generative AI, LLM, GPT4, Falcon, Github, Fine-Tuning, Lang chain, API

1 Background

The proliferation of open-source projects on platforms like GitHub has indeed contributed to a vast repository of textual documentation, a crucial resource for comprehending and utilizing software repositories. README files, in particular, stand as the primary gateway to these projects, encapsulating essential information that significantly influences initial engagement and ongoing usage of the repository. However, despite their importance, READMEs face inherent challenges that can limit their effectiveness, primarily rooted in users' diverse preferences for content consumption and the difficulties associated with processing large volumes of text.

One of the notable limitations of README files on platforms like GitHub is their potential to be uninsightful and uninteractive. Traditional READMEs often rely on lengthy paragraphs and technical jargon, making it challenging for users to quickly grasp the key concepts and functionalities of a project. This lack of clarity can lead to frustration and disengagement, especially for users who prefer more visual and interactive learning experiences.

The need for more visual aids in understanding README files is underscored by the fact that the human brain processes visual information significantly faster than text approximately 60,000 times faster. Visual elements, such as images, diagrams, and interactive media, can convey complex information more efficiently and intuitively. Unlike plain text, visual aids have the power to break down intricate concepts, providing a clearer understanding of the project's structure, functionality, and potential use cases.

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task[1]. Moreover, the diverse backgrounds and skill levels of users navigating open-source projects on platforms like GitHub necessitate a more inclusive approach to documentation. Visual aids can bridge the gap between novices and experienced developers, offering a more accessible entry point into the intricacies of a project. By incorporating visual elements, README files can cater to a broader audience, accommodating different learning styles and levels of technical expertise.

In conclusion, while README files play a vital role in disseminating information about open-source projects, their effectiveness can be compromised by textual limitations and varying user preferences. The demand for more visual aids arises from the recognition that visual information is processed more rapidly and comprehensively by the human brain. Incorporating visual and interactive elements in READMEs not only enhances accessibility but also ensures a more engaging and user-friendly experience, contributing to the broader goal of making open-source projects more inclusive and comprehensible for all users.

CSC 591, GenAI4SE, 2023, NC State

ACM ISBN 2023
https://github.ncsu.edu/ncheruk2/Readme2Video

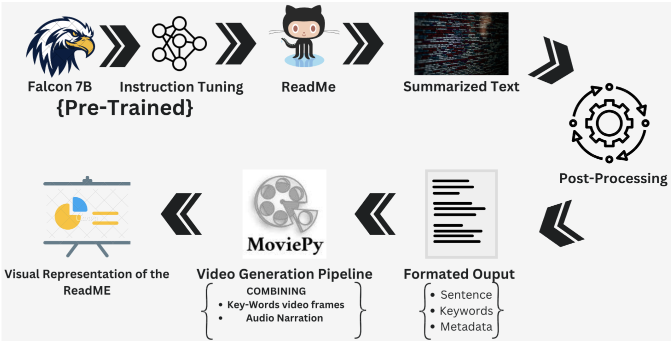


Figure 1. The Workflow

2 Problem Definition

Despite the critical role of README files, they present several challenges:

1. **Information Overload:** Users are often faced with dense and lengthy READMEs that can be overwhelming, leading to disengagement.
2. **Learning Styles:** Textual content does not cater to all learning styles, particularly visual and auditory learners who may find videos more effective.
3. **Engagement:** Text lacks the dynamic elements of video content, which can more effectively capture and retain user attention.
4. **Time Efficiency:** Users may require quick overviews of repositories without delving into detailed documentation, which text does not easily afford.

3 Project Objective

The ReadMe2Video project aims to address these challenges by:

1. Summarization: Utilizing generative AI to create concise summaries of README files, ensuring only pertinent information is presented.
2. Keyword Extraction: Identifying keywords from summaries to capture the essence of the repository's content.
3. Video Synthesis: Finding and stitching together relevant video clips based on extracted keywords to create an engaging narrative of the README content.
4. Accessibility Enhancement: Making the information in README files more accessible to a broader audience by transforming text into a more digestible video format.

4 Literature Survey

- **Categorizing the Content of GitHub README Files:** This study focuses on the importance of README files in GitHub repositories, emphasizing their role in shaping the first impression of a software project. The researchers conducted a qualitative analysis of 4,226 README file sections from 393 GitHub repositories. They developed a classifier to automatically categorize these sections, finding that information about the 'What' and 'How' of a repository is commonly included, while many READMEs lack details on the purpose and status of the repository. The study also highlights the potential of automated classification in enhancing the quality of documentation on GitHub, making it easier for developers to discover relevant information[2].

- **Make-A-Video: Text-to-Video Generation without Text-Video Data:** The research paper focuses on creating short video clips from textual inputs through a blend of generative adversarial networks (GANs) and convolutional variational autoencoders (CVAEs). This study tackles the complexities of combining text and static background information to produce dynamic video content. Such methodologies are highly pertinent to projects that aim to generate videos that are not only captivating but also accurately represent the textual content they are based on.
Additionally, the "Make-A-Video" paper presents a novel approach for text-to-video (T2V) generation. It innovatively utilizes image priors and operates without the need for text-video data pairs, overcoming a significant limitation seen in previous studies. This method builds on a text-to-image (T2I) model, enhancing it with spatiotemporal convolution and attention layers. This approach is particularly crucial for projects focused on generating videos directly from textual descriptions, addressing a core aspect of such initiatives[3].

- **Hybrid Long Document Summarization Using C2F-Far And ChatGPT: A Practical Study:** Combines C2F-FAR's ability to extract important sentences from texts and ChatGPT's zero-shot ability to summarize and paraphrase texts. The results measured against the existing automated evaluation metrics are as good as those written by humans, especially when the ROUGE-1 score is taken into account. Limitations: Critical issues were identified in the human evaluation in terms of text coherence, faithfulness and style. ChatGPT is not open source

- **Recent Video-based Learning Research:** A study conducted a longitudinal analysis on the impact of video-based learning in business statistics. This study collected historical datasets spanning several years to investigate student achievement before and after the introduction of VBL. The findings from this research provides insights into how video-based learning affects student outcomes over time[4]

5 Methodology

5.1 Dataset Collection

Motivated by the absence of an readily available dataset for our specific problem statement, we undertook the initiative to curate our own collection. In total, we meticulously gathered 225 readme files associated with various courses.

To curate a comprehensive dataset of course-related readmes, a systematic process has been developed. This process involves a series of steps to ensure the inclusion of relevant information, whether extracted from existing readmes or generated from course description materials.

Step 1: Search for Course-Related Readmes

To initiate the dataset collection process, an exhaustive search was conducted across various online platforms, repositories, and college websites. The goal is to identify readmes associated with different courses. This step involves leveraging search engines and specialized platforms such as GitHub to locate relevant readme files.

Step 2: Add Readmes to the Dataset Stack and Summarize with GPT-4

Upon identifying course-related readmes, they were added to the dataset stack. To extract meaningful information from these readmes, the contents were summarized using the advanced natural language processing capabilities of GPT-4. This involves feeding the readme content into the model and extracting concise yet informative summaries that capture the essence of each document.

Step 3: Generate Readmes from Course Description Materials

In cases where specific readmes for courses are not readily available, the process seamlessly transitions to utilizing course description materials. These materials, which may exist in various formats such as PDF or DOC files, are converted into readme files.

The conversion process involves extracting key details from the course description, such as objectives, prerequisites, and outcomes. This information is then structured into a readme format that provides a clear overview of the course content.

Step 4: Add Generated Readmes to the Dataset Stack and Summarize with GPT-4

The newly created readmes, generated from course description materials, were added to the dataset stack. Similar to Step 2, GPT-4 is employed to summarize the contents of these readmes. This ensures that the dataset maintains a consistent level of abstraction and summarization across all included documents.

By following this elaborate dataset collection process, a robust compilation of course-related readmes was achieved. The combination of manual curation, machine-generated summaries, and dynamic readme creation from course materials ensures a diverse and informative dataset for future applications in education and natural language processing research.

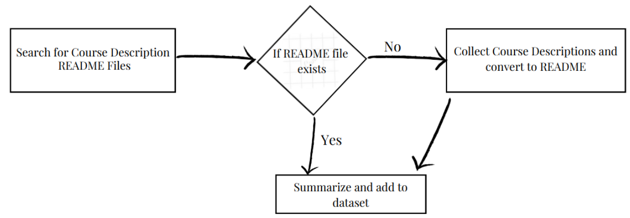
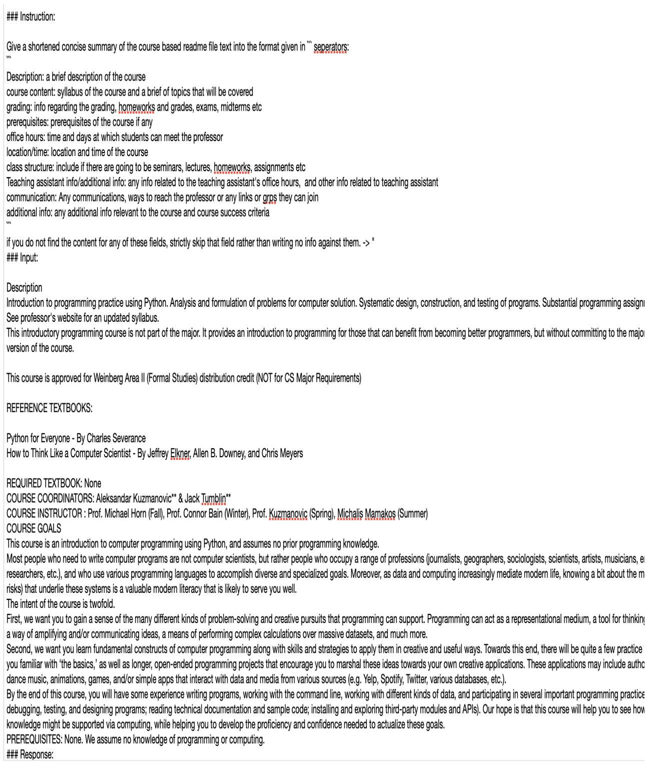


Figure 2. Process of Data Collection.

5.2 Instruction tuning

We have taken off-the-shelf falcon-7b from hugging face and used the Trainer module to finetune it with our custom instruction format dataset. Falcon-7B is a 7B parameters causal decoder-only model built by TII and trained on 1,500B tokens of RefinedWeb enhanced with curated corpora. It is made available under the Apache 2.0 license. We have chosen this model because it outperforms comparable open-source models (e.g., MPT-7B, StableLM, RedPajama etc.), as is trained on 1,500B tokens of RefinedWeb enhanced with curated corpora. See the OpenLLM Leaderboard. It features an architecture optimized for inference, with FlashAttention (Dao et al., 2022) and multiquery (Shazeer et al., 2019).

- We had 225 quality data sample that we prepared as explained in the previous section
- Then we converted each data pair(complete readme text and corresponding summary) to instruction format as shown in below figure.
- Then we loaded the pre trained Falcon 7B model in fp16 precision as it was observed that finetuning a quantized model rather than a full precision model wouldn't decrease the model performace drastically.
- Also we reduce our memory footprint almost by half by loading the model in half precision.
- Converted lm_head and small parameters to fp32 for stability and trained LoRA adapters on it.
- We connected our model training to weights and biases for real time monitoring of our training loop.
- LoRA is an improved finetuning method where instead of finetuning all the weights that constitute the weight matrix of the pre-trained large language model, two smaller matrices that approximate this larger matrix are fine-tuned. These matrices constitute the LoRA adapter. This fine-tuned adapter is then loaded to the pretrained model and used for inference.
- Our model reached convergence after 260 epochs(as seen in Figure 1) with a final loss value of 1.274.



```
### Instruction:

Give a shortened concise summary of the course based readme file text into the format given in ``` seperators:
```
Description: a brief description of the course
course content: syllabus of the course and a brief of topics that will be covered
grading: info regarding the grading, homeworks and grades, exams, midterms etc
prerequisites: prerequisites of the course if any
office hours: time and days at which students can meet the professor
location/time: location and time of the course
class structure: include if there are going to be seminars, lectures, homeworks, assignments etc
Teaching assistant info/additional info: any info related to the teaching assistant's office hours, and other info related to teaching assistant
communication: Any communications, ways to reach the professor or any links or grps they can join
additional info: any additional info relevant to the course and course success criteria
```
if you do not find the content for any of these fields, strictly skip that field rather than writing no info against them. -> "
### Input:

Description
Introduction to programming practice using Python. Analysis and formulation of problems for computer solution. Systematic design, construction, and testing of programs. Substantial programming assign
See professor's website for an updated syllabus.
This introductory programming course is not part of the major. It provides an introduction to programming for those that can benefit from becoming better programmers, but without committing to the majo
version of the course.

This course is approved for Weinberg Area II (Formal Studies) distribution credit (NOT for CS Major Requirements)

REFERENCE TEXTBOOKS:

Python for Everyone - By Charles Severance
How to Think Like a Computer Scientist - By Jeffrey Elkner, Allen B. Downey, and Chris Meyers

REQUIRED TEXTBOOK: None
COURSE COORDINATORS: Aleksandar Kuzmanovic** & Jack Tumblin**
COURSE INSTRUCTOR : Prof. Michael Horn (Fall), Prof. Connor Bain (Winter), Prof. Kuzmanovic (Spring), Michalis Mamakos (Summer)
COURSE GOALS
This course is an introduction to computer programming using Python, and assumes no prior programming knowledge.
Most people who need to write computer programs are not computer scientists, but rather people who occupy a range of professions (journalists, geographers, sociologists, scientists, artists, musicians, e
researchers, etc.), and who use various programming languages to accomplish diverse and specialized goals. Moreover, as data and computing increasingly mediate modern life, knowing a bit about the m
risks) that underlie these systems is a valuable modern literacy that is likely to serve you well.
The intent of the course is twofold.
First, we want you to gain a sense of the many different kinds of problem-solving and creative pursuits that programming can support. Programming can act as a representational medium, a tool for thinkin
a way of amplifying and/or communicating ideas, a means of performing complex calculations over massive datasets, and much more.
Second, we want you learn fundamental constructs of computer programming along with skills and strategies to apply them in creative and useful ways. Towards this end, there will be quite a few practice
you familiar with 'the basics,' as well as longer, open-ended programming projects that encourage you to marshal these ideas towards your own creative applications. These applications may include autho
dance music, animations, games, and/or simple apps that interact with data and media from various sources (e.g. Yelp, Spotify, Twitter, various databases, etc.).
By the end of this course, you will have some experience writing programs, working with the command line, working with different kinds of data, and participating in several important programming practice
debugging, testing, and designing programs; reading technical documentation and sample code; installing and exploring third-party modules and APIs). Our hope is that this course will help you to see how
knowledge might be supported via computing, while helping you to develop the proficiency and confidence needed to actualize these goals.
PREREQUISITES: None. We assume no knowledge of programming or computing.
### Response:
```

Figure 3. Instruction data format

Figure 4. Loss converging

5.3 Abstractive Summarization of readme

We use our finetuned LLM with Lang chain to generate coherent summaries. There are different ways to summarize a document using LLM's.

Stuff Method:

- It "stuffs" text into the prompt as context in a way that all of the relevant information can be processed by the model at once.
- Advantage: Only requires a single call to the LLM, which is faster than other methods that require multiple calls. When summarizing, LLM has access to all the data at once, which can result in a better summary.
- Disadvantage: Depending on the context length of LLM, the stuff method would not work as it result in a prompt larger than the context length.
- In our case for small to medium-sized readmes directly input the readme text into the prompt so that the model has access to the entire context. We are able to generate coherent summaries with this method.

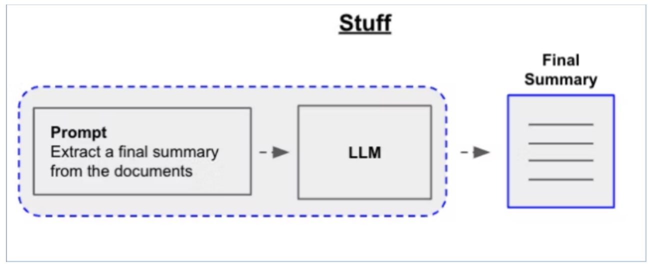


Figure 5. Summary generation pipeline using stuff chain

Map Reduce Method:

- In MapReduce method we implement a multi-stage summarization. It is a technique for summarizing large pieces of text by first summarizing smaller chunks of text and then combining those summaries into a single summary.
- The map-reduce chain breaks the document down into context length token chunks max. Then it runs the initial map prompt you define on each chunk to generate a summary.
- then use a combine prompt to generate a collated summary of the chunk wise summarizations.
- Advantage: With Map reduce method, the model is able to summarize a large paper by overcoming the context limit of Stuffing method with parallel processing.
- Disadvantage: MapReduce requires multiple calls to the model and potentially losing context between chunks.
- In our case the results with this method are half baked and sub optimal. It might be because we have fine-tuned a 7B model and optimal context parsing happens smoothly with much larger models.

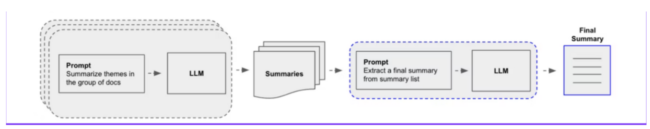


Figure 6. Summary generation pipeline using map reduce chain

6 Findings

1. Keyword Generation Challenges:

One of the key findings in our project pertains to the challenges associated with keyword generation from each sentence within the readme files. We observed that the relevance of generated keywords was not consistently apt. This raised concerns about the precision of the summarization process, prompting us to explore alternative approaches for improving the accuracy of keyword extraction.

2. Heavy Reliance on Image API's Database:
In the context of creating video clips associated with specific keywords, our project revealed a notable dependence on the Image API's database. The success and effectiveness of these clips were found to be closely tied to the richness and diversity of the Image API's image repository. This discovery underscores the importance of optimizing and diversifying the image database to enhance the overall quality and engagement of the generated video content.

3. Efficient Refinement of Falcon Base on a Custom Dataset:
A noteworthy outcome of our project involves the successful refinement of a pre-trained open-source Falcon base. By leveraging a high-quality custom dataset, even with a limited number of samples, we achieved commendable results in terms of efficiency. This underscores the potential effectiveness of custom datasets in fine-tuning pre-existing models for specific tasks, showcasing the adaptability and robustness of our approach.

In summary, our findings shed light on the intricacies of keyword generation, the crucial role of the Image API's database in video clip creation, and the efficiency gained through the refinement of a Falcon base with a tailored dataset. These insights contribute to the ongoing enhancement of our project, ensuring a more engaging and accessible transformation of readme files into interactive videos for a diverse audience.

7 Video Pipeline

7.1 Videos from Pexels API

The video processing pipeline is integral to the project, orchestrating the synthesis of a coherent visual narrative. Its inception involves assimilating the generated summary from the model alongside corresponding keywords to construct a video clip by clip. Our utilization of the Pexels API facilitates the targeted retrieval of videos associated with specific keywords, optimizing relevance. Subsequently, these videos are meticulously stored locally to streamline the production process.

7.2 Audio from ElevenLabs

The audio segment is meticulously crafted using the sophisticated capabilities of the ElevenLabs Text to Speech service. This service, chosen for its advanced features, excels in transforming textual content into audio with remarkable naturalness and lifelike quality. The initial phase entails the generation of audio files, pivotal as foundational components shaping subsequent stages of video creation. Parameters such as narration duration are intricately considered during this stage to ensure optimal audio coherence.

7.3 Text from Summarized Model Output

Simultaneously, the language model contributes to the content creation process by generating a concise and coherent textual summary based on the input data. This summarized text serves as a textual backbone for the ensuing audio and visual elements, ensuring thematic consistency and relevance throughout the video.

7.4 Combining with MoviePy

The convergence of these distinct yet interconnected components—sentences, video, and audio—is orchestrated seamlessly using the MoviePy library. This Python library empowers us to intricately combine and concatenate individual video clips. Additionally, MoviePy facilitates the integration of a carefully curated background score, enhancing the overall aesthetic appeal of the final video composition. Following this, the assembled video undergoes rendering and is efficiently stored in the local repository[6].

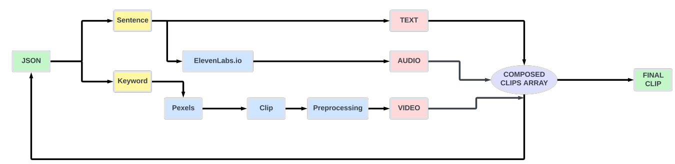


Figure 7. Video Pipeline

8 Evaluation

8.1 Rationale for Human Evaluation

Human evaluation is pivotal in assessing the effectiveness of AI-generated videos from GitHub README files. The primary aim of the ReadMe2Video project is to enhance user engagement and comprehension through visual summaries of textual documentation. Given that the end-users of such videos are humans with diverse preferences and cognitive processes, it is essential to understand their subjective experiences and judgments of the content.

8.2 Objectives of Human Evaluation

1. Comprehensiveness: Evaluating whether the video summaries include all critical information from the README files.
2. Relevance: Assessing if the keywords extracted are representative of the main topics and if the video content aligns with the summarized information.
3. Clarity: Determining if the video presentation helps in better understanding the README content compared to text.

4. Engagement: Gauging the ability of the videos to capture and retain the viewers' attention and interest.

8.3 Methodology

The evaluation will be conducted using a Google Form that will present participants with three different types of videos:

1. Videos generated by the project's algorithm.
2. Videos generated using a summarization tool like ChatGPT.
3. Videos created using an online video generation tool.

Participants will be asked to rate each video on:

- The accuracy and quality of README summarization.
- The relevance of the keywords extracted.
- The relevance of the video content to the topic of the README.

8.4 Calculation of Evaluation Score The scores from each category will be averaged to produce a composite score for each video. This quantitative data will provide a clear metric for comparing the performance of the different video generation methods.

8.5 Importance of Human Evaluation Human evaluation is crucial for several reasons:

1. Subjective Experience: AI metrics may not capture the subjective user experience effectively, especially concerning engagement and comprehension.
2. Qualitative Insights: Qualitative feedback from users can provide insights into the nuances of their preferences, which are not always quantifiable.
3. Algorithm Validation: Human judgment is required to validate the relevance and effectiveness of the algorithms used for summarization and keyword extraction.
4. Iterative Improvement: The feedback will guide iterative improvements to the video generation process, ensuring that the final product meets user expectations.

	GPT	Falcon	InVideo
Text Summarization	4.5	4.0	-
Keywords	4.0	3.5	-
Overall Video Generated	4.3	3.9	3.8

Table 1. Evaluation across different Models

8.6 Evaluation Conclusion

In alignment with the project's goal of making technical information more accessible and engaging, a human-centric evaluation approach has been adopted for the Falcon 7b instruct, which underwent fine-tuning. This evaluative strategy involves incorporating direct user feedback into the ReadMe2Video project, allowing for the refinement of its algorithm to better cater to the needs of the GitHub community. Notably, the Falcon 7b instruct achieved an impressive BLEU score of 0.417 after fine-tuning, reflecting its enhanced performance and accuracy in language processing tasks. This evaluation not only serves to validate the effectiveness of the AI-generated videos but also identifies areas for improvement, propelling the project toward its overarching objective of revolutionizing the accessibility of open-source project documentation.

9 Lessons Learnt

9.1 Importance of Model Parameters in Summary Quality One key lesson learned from our project is the critical role that model parameters play in determining the quality of the inference summary. The effectiveness of the summarization process is significantly influenced by the parameters used in both the fine-tuning and inferencing stages of the model. It was observed that selecting the appropriate parameter set is crucial for producing high-quality output. Experimentation with different parameter configurations and a nuanced understanding of their impact on summary generation is essential to achieve optimal results. This insight emphasizes the need for a careful and iterative approach in parameter selection to enhance the overall performance of the model.

9.2 Human Evaluation as an Effective Assessment Method

In the context of our domain-specific use case, human evaluation emerged as one of the most effective methods for assessing the quality of the generated summaries. While automated metrics provide valuable quantitative insights, human evaluation adds a qualitative layer of understanding. The nuanced nature of README files, with domain-specific terminology and contextual intricacies, makes human judgment particularly valuable. By soliciting feedback from human evaluators, we gained a deeper understanding of how well the generated summaries align with user expectations, ensuring that the content remains accurate, informative, and contextually relevant. This lesson underscores the importance of incorporating human perspectives to validate and refine the output of automated systems, especially in specialized domains.

In summary, the project highlighted the pivotal role of model parameters in determining the quality of inference summaries. It also underscored the effectiveness of human evaluation as a crucial assessment method, particularly in domain-specific use cases. These lessons contribute to a more nuanced and informed approach in fine-tuning and evaluating models for the transformation of README files into engaging and accessible interactive videos.

10 Conclusion

10.1 Commendable Video Generation for Course Syllabus READMEs:

Our project's pipeline has successfully demonstrated the capability to generate commendable videos, providing an effective interface for learning within the context of a given course syllabus's specific README. By translating textual information into engaging visual content, our system enhances the accessibility and comprehensibility of course materials. This finding underscores the potential of our approach to transform traditional README files into dynamic and visually appealing learning resources.

10.2 Large Language Model for Text Summarization and Keyword Generation:

The core of our success lies in the utilization of a Large Language Model (LLM) for text summarization and keyword generation. This approach has proven to be particularly suitable for application-based, domain-specific pipelines, such as video generation for educational content. Leveraging the advanced linguistic capabilities of the LLM, we have effectively condensed lengthy READMEs into concise summaries while extracting relevant keywords. This not only streamlines the video creation process but also ensures that the generated content remains contextually rich and informative.

In conclusion, our project's achievements highlight the transformative potential of employing a Large Language Model for converting README files into interactive videos. The commendable video generation and the adaptability of our approach for domain-specific pipelines emphasize the broader applications of this technology in making educational materials more engaging and accessible to a diverse audience.

10.3 Code and Videos

The videos are stored in the drive - Google Drive
We have used GitHub to store the code - GitHub

11 Future Work

11.1 Streamlining Keyword Fine-Tuning with Image Search:

One crucial avenue for future work involves streamlining the process of keyword fine-tuning by integrating image search functionalities. By combining keyword extraction with relevant visual elements sourced from image databases, the goal is to construct a high-quality, domain-specific clips dataset. This enhancement aims to further enrich the video content, aligning it more closely with the nuanced nature of the original README files. Improved synergy between textual and visual elements could contribute to a more comprehensive and engaging representation of the underlying information.

11.2 Soliciting Human Feedback for Video Evaluation:

In the pursuit of refining and optimizing video generation, seeking human feedback becomes paramount. Future work includes the solicitation of feedback for the three distinct types of videos generated in the project—Baseline, GPT summary, and Falcon summary. Human evaluations can provide valuable insights into the perceived effectiveness, clarity, and engagement levels of each video type. This iterative feedback loop ensures that the end product aligns more closely with user expectations and preferences, ultimately enhancing the overall quality and impact of the generated videos.

11.3 Concluding the End-to-End Pipeline for Web Application Hosting:

A crucial step towards making the project a functional tool involves concluding the end-to-end pipeline, encompassing both the video and summarization pipelines, for web application hosting. This entails integrating the developed system into a user-friendly web application interface. Users should be able to easily upload their README files, receive summarized content, and interact with dynamically generated videos. The focus on web application hosting transforms the project from a research endeavor into a practical and accessible tool, ensuring that the benefits of enhanced README accessibility reach a broader audience.

The outlined future works aim to refine and extend the capabilities of the project. From augmenting the dataset with visual elements to leveraging human feedback for evaluation and finalizing the end-to-end pipeline for web application hosting, these steps collectively contribute to the project's evolution into a robust and user-centric tool for making README files more engaging and accessible to a diverse audience.

References

[1] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. *ArXiv*, https://arxiv.org/abs/2005.14165.

[2] Prana, G. A., Treude, C., Thung, F., Atapattu, T., & Lo, D. (2018). Categorizing the Content of GitHub README Files. *ArXiv*, https://arxiv.org/abs/1802.06997.

[3] Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., Parikh, D., Gupta, S., & Taigman, Y. (2022). Make-A-Video: Text-to-Video Generation without Text-Video Data. *ArXiv*, https://arxiv.org/abs/2209.14792.

[4] Navarrete, E., Nehring, A., Schanze, S., Ewerth, R., & Hoppe, A. (2018). A Closer Look into Recent Video-based Learning Research: A Comprehensive Review of Video Characteristics, Tools, Technologies, and Learning Effectiveness. *ar5iv*, https://ar5iv.labs.arxiv.org/html/2301.13617.

[5] Lu, G., Larcher, S. B., & Tran, T. (2023). Hybrid Long Document Summarization using C2F-FAR and ChatGPT: A Practical Study. *ArXiv*,

arXiv:2306.01169.
[6] Deepa, R., & Lalwani, K. N. (2019). Image Classification and Text Extraction using Machine Learning. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India (pp. 680-684). doi: 10.1109/ICECA.2019.8821936.