**Music Video Production with Generative AI**

Wang Ruixiang

Li Hensheng

Zhong Minjun

Fernando Ramírez González

Technische Universität München

NLP-Projekt Woche

Prof. Dr. Veronika Gamper

18. Juli 2024

**Music Video Generation with Generative AI**

*Introduction*

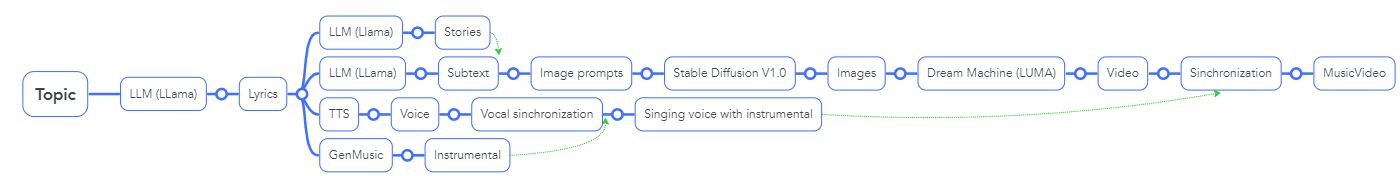
The use of AI (Artificial Intelligence) supported systems has seen a strong increase in recent years due to its positive impact in productivity, allowing workers, artist and companies to reduce the resources and amount of time needed to create text, code and images, needed for most of today’s digital businesses, for the major part the technology that allows these productivity boost is based on NLP (Natural Language Processing), which focuses on extracting relevant insights from text information, or on Generative AI, which leverages the power of statistical modelling learn different characteristics of the input data and produce further

One of its most recent and interesting use cases, is the creation of musical videos, for this task the power of generative AI can be used, still due to is complexity the task needs to be broken down into 4 main steps:

1. Lyric and text description generation
2. Video/Image generation
3. Music generation
4. Integration of all individual modules

For this reason, we decided to introduce the following workflow

Version 1.0

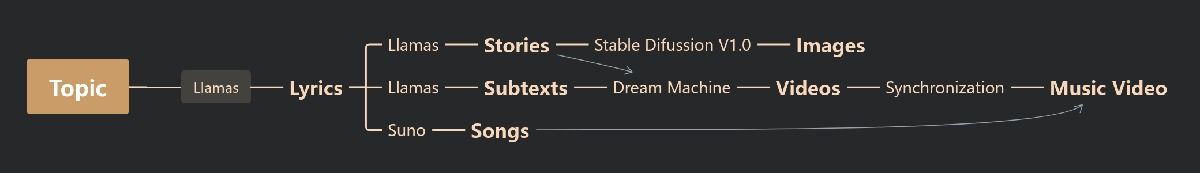
From the task breakdown and with the first version of the workflow the code was tested into individual modules:

1. LLM manager: The purpose of this class was to establish access to the LLama model via the Together API, together with a collection of prompt templates and regex functions that allowed to automate the
2. create some prompt templates basic lyric prompts and templates that allo

LLM model a class called LLMManager was created, the class can be described as follows:

Input:

Workflow:



**Input:**

Take "I met my ex on Tik-Tok " as the topic, and that is the only input we need

**1.step: Topic to lyrics**

We use llamas Api key to process the topic and convert it to lyrics.

**2.step:** **lyrics to Stories**

To make the video more connected to the song, we convert the lyrics to Stories by using the prompt:” As a story writer, your task is to create a short story based on a given song lyric snippet. Each story should be detailed, highlighting the characters' emotions and relevant actions, and must be closely related to the content of the song lyrics. Each story should be at least 200 words long. Your response should vividly capture the essence of the song lyrics, incorporating the emotions and actions of the characters in a way that resonates with the given snippet. The story should convey a strong connection to the lyrical content, enriching the narrative with depth and relevance. Please ensure that each story is crafted with attention to detail and creativity, immersing the reader in a compelling and meaningful tale that aligns closely with the song lyric snippet provided. The song lyric snippet is as follows: {lyrics}”

**3.step:text/images to videos**

Problem:

1. In **first pipeline we** need to use Stable Diffusion to generate images, which is also called key frames, and use Deam Machine Luma to generate multiple videos. But the short videos generated images does not work well. Below are examples.



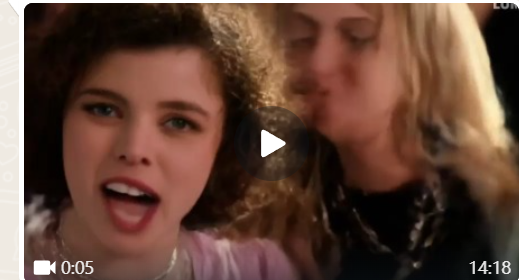
Then we try to rewrite the prompt. Now we can find the generateImagePrompts(lyrics) function in src\ LLMProcessing\LLM\_processing.py. The input is lyrics, and the output is image\_prompts\_list, and we use furtherImprovePrompt (image\_prompt). replace ('\n', '').strip() to improve the image\_prompt ,which is the input of the function dreamMachineMake(image\_prompt, access\_token, img\_file). Then we can generate multiple short videos by only running main.py.

2.The reason we did not merge the furtherImprovePrompt(image\_prompt) and generateImagePrompts(lyrics) into one function because the code is added to make the dream machine understand prompt. And the input: prompt is the output of the function generateImagePrompts.

def furtherImprovePrompt(self, prompt):

return self.ask\_llama\_3\_8b\_TOGETHER\_API (f"Improve the following scene by eliminating irrelevant details and improving engagement quality, maintain the same format: {prompt}")

3.The below pictures show the video generated by text without negative prompt.





We can see that the front character is fine, but the features of the back character are distorted.Then we shortened positive prompt to emphasize key features and used Negative prompt: bad anatomy, bad proportions, blurry, cloned face, deformed, disfigured, duplicate, extra arms, extra fingers, extra limbs, extra legs, fused fingers, gross proportions, long neck, malformed limbs, missing arms. And successfully fix the problem.

Result:

We only need the prompts of key frames to generate videos.

**4.step: lyrics to song**

Problem:

Result:

We use suno to generate the songs and save locally.

**5.step:** Synchronize video and audio

Problem:

Because one account of Luma can only generate 5 short videos per day, it made the debug process terribly slow. Then we failed to Synchronize video and audio.

Result:

Synchronize video and audio manually.

**6.step: integration of all modules**

Problem:

Suno and Luma do not give the api key to users. Fortunately Li has found a “leak” on Github for using the api key for free.

Result:

We

Important Functions:

**def uniteTags(text, LLManager)**

The main purpose of the function is to extract specific music tag information from the given text. These tags include tempo, key, audio recording, vocal performance, and time signature. This information is obtained by calling methods of the LLManager object, with each method receiving the text as an argument and returning the corresponding information.

After obtaining all the necessary information about the music labels, this information is spliced into a string, with the parts separated by spaces. Finally, this spliced string is returned.

**async def download\_video(url, filename):**

Download the video from the given url and save it to the specified filename

**First pipeline: async def process\_topic(topic, LLManager, diffusionManager, access\_token):**

The main objective is to generate a final video with audio and images from a given topic. Below is a detailed explanation of the code:

1. Initialization and Text Generation:

We use the function LLManager.generateText to generate text content based on the topic, and tags = uniteTags(text) to get the tags, then we initial an empty list to store short videos.

2.Audio generation and check:

Build the request load payload and call custom\_generate\_audio to generate audio data. Then get the ID of the generated audio and print it using the loop. At last, we check the status of the audio until it changes to "streaming". Once all done, the process will automatically download the audio file and save it locally.

3. Generate Images and Keyframes:

Generates a list of image prompts and keyframes. For each keyframe and corresponding image prompt, generates an image by using Stablediffussion and saves the image file.

4.Generate video using dreamMachineMake:

Calls the dreamMachineMake function to create a video generation task then downloads the generated video files and adds them to video\_list.

5. Merge All Generated Videos:

By using the moviepy.editor module we can easily merge all generated video clips into one final video. At last, we merge the audio clips with the video to produce the final output file.

**Second pipeline: async def process\_topicCompleteVideo(topic, LLManager, diffusionManager, access\_token):**

The main objective is to generate a final video with audio and texts from a given topic. The difference from the function processtopic(topic, LLManager, diffusionManager, access\_token) is it generates the videos by using texts of keyframe, which is the same as the prompts of image prompts in that function.

**async def main ():**

It allows us to give the access\_token of luma and topics we want to try in there. And call the function of first or second pipeline.

**Reference**

[1] [GitHub - mikezzb/lyrics-sync: A deep learning lyrics-to-audio alignment system, generating synchronized lyrics from a song and its lyrics](https://github.com/mikezzb/lyrics-sync?tab=readme-ov-file)