International Journal of Engineering Applied Sciences and Technology, 2024
Vol. 8, Issue 12, ISSN No. 2455-2143, Pages 234-237
Published Online April 2024 in IJEAST (http://www.ijeast.com)

DEVELOPMENT AND IMPLEMENTATION OF VIDEO LANGUAGE TRANSLATOR USING PYTHON LIBRARIES FOR GLOBAL COMMUNICATION

Deepanshu Koli, Anmol Singal, Dr. Amrita Goel, Dr. Vasudha Bahl, Ms. Nidhi Sengar
Maharaja Agrasen Institute of Technology
Information Technology and Engineering

Abstract: **This Research paper examines specific development and implementation of video language interpretation, and uses Python libraries for rigorous analysis. The main objective of this new tool is to extract audio content from a given video, translate the extracted audio into the language of the target location, then recombine the translated audio as an exclusive video broadcast should change the format, thereby putting information in the hands of a global target audience.**

Keywords: **Python, MoviePy, Spleeter, Whisper Timestamped Model, pyttsx3, pydub, flask**

I. INTRODUCTION

In the complexity of multimedia communication, the challenge of overcoming language barriers in video content emerges as a difficult effort. This study uses the skills of Python libraries to examine the challenges of multimedia interpretation, delving into the specific development and precautions of video language interpreters. The main purpose of this tool is to seamlessly navigate through the languages of video content. By extracting audio from a given video source, the system attempts to overcome language limitations with an advanced
rendering system. The decoded audio is then seamlessly reintegrated into the original video, imagining changes that would change the global distribution of information. As technology continues to evolve, the need for more versatile and effective solutions to prevent linguistic fragmentation in multimedia becomes more apparent This development attempts to address this need through micro capabilities of customized Python libraries for analysing multimedia applications. The video language translator aims to empower content creators to reach a global audience with unprecedented ease.

II. BACKGROUND

In the rapidly evolving landscape of multimedia communication, the convergence of technologies and languages presents both challenges and opportunities. Canvas videos often face formidable language barriers, making them inaccessible to a global audience. [1]
As multimedia content continues to flourish, the need to transcend language boundaries becomes important. The complexity of this task is tackled by leveraging the power of Python libraries, presenting a careful and versatile approach to multimedia interpretation. We focus on developing and deploying video language interpreters functionally, which attempts to complicate language barriers in video content. The main objective of the proposed tool is ambitious but flexible. Making full use of Python libraries such as MoviePy, Spleeter, Whisper Timestamped Model, pyttsx3, and pydub, we aim to develop a solution that easily extracts audio from videos, translates them into audio transmissions language of the target audience, and it recombines the decoded audio with the original video. This visionary approach envisions a worldwide spread of information unhindered by language barriers.
With multiple languages, state-of-the-art technologies, and dynamic multimedia communication, this research stands as a beacon when content producers want to reach a global audience and the proposed video translator shows up as a potential game changer. The integration of standard Python libraries in this effort not only demonstrates technical prowess but positions this research as an important contribution to the evolving field of multimedia translation and accessibility.[2]

III. METHOD

1. Audio extraction and translation

a. video_to_audio

Purpose: Extract voice from video input and save it for interpretation.

Libraries used:

• Subprocess for executing external commands.
• Moviepy for video processing. • Spletter provides audio

separation in the form of adding voice.
Approach: Use the Spleeter to remove voice and extras from the video input. Save the voice as a separate audio file.[4]

b. audio_to_srt
Purpose: Translate the extracted audio into the language of your choice.
Libraries used:
• Subprocess for executing external commands.
• whisper_timestamped for audio translation
Approach: Use the Whisper program to decode the extracted audio. Save the decoded audio as a file.[5]

2. <u>Subtitle generation from rendered audio</u>
a. srt_to_audio
Purpose: Create subtitles from rendered audio.
Libraries used:
• pysubparser for parsing SRT files. • pyttsx3 For text-to-speech
integration.
• pydub for audio processing.

Approach: Use the pysubparser library to parse the decoded audio into subtitles. Put the material together into a talk for each subheading. Combine the audio parts to create the final audio file.[7]

3. <u>Video Recombination</u>
a. video-audio
Purpose: Configure the entire
functionality by integrating the
functionality of the above modules.

Libraries used:
• os For operating system services.
• shutil for file-directory operations.
• moviepy editing videos. [3]
• Other custom modules

Approach:
• Add music to video input using video_to_audio.py.
• Use audio_to_srt.py to translate the video audio to the language of your choice.
• Create audio from defined subtitles using srt_to_audio.py.
• Combine the input video with the decoded audio to produce the final video output.

b. ISO Mapping
Purpose: Provide a mapping between user-friendly language names and ISO grammar rules. [11]
Approach: Use a dictionary
(language_mapping) to map user-friendly language names to ISO language codes.

4. <u>Web Interface</u>
Purpose: Acts as a user interface for uploading and starting processing of videos.
Libraries used:
• Flask for web development. [8] • Flask-SocketIO provides real-time communication between server and client. [9]
Approach: Allow users to upload a video file and select the output language. Use SocketIO to issue additional enhancements at the beginning of the process.

5. <u>Cleanup</u>
Purpose: Repair temporary files and directories created during operation.
Approach: Use the cleanup_files function to remove temporary files and directories.

6. <u>Example implementation</u>
Purpose: Demonstrate how to use the program to process a sample video file.

Approach:
• Set the video channel.
• Add music to video, translate audio, create audio from notes, and merge video and audio.
• Delete the temporary files after processing.

The Method combines audio processing, rendering and video reconnection using several Python script libraries. This comprehensive approach accomplishes the ultimate goal of creating multilingual video for a global audience by breaking language barriers. The integrated Python libraries present a complete solution for video language interpretation.
The web interface provides a user-friendly way to interact with the tool, making it accessible to a wider audience.

Topic Analysis
<u>Multimedia processing :</u>
• **Video-to-Audio Extraction**: The code uses the MoviePy library to extract audio from Video files (video_to_audio.py).
• **Audio translation**: The Audio_to_srt.py module uses the Whisper API for audio translation, converting the extracted audio to the desired language.
• **Audio-video recombination:** Using MoviePy, the translated audio is added to the original video, creating a video playback (main.py). [3]

<u>Language translation and mapping:</u>
• **Whisper API for Audio Translation:**
The audio_to_srt.py module integrates the Whisper API for audio content translation, allowing for precise language translation.[13]

• **Language mapping:**
the mapping_language.py module maps the name of the destination language to an ISO code, simplifying language identification during the translation process.

SRT (SubRip Subtitle) Integration:
• **Subtitle parsing:**
The srt_to_audio.py module parses SRT subtitle files, extracting captions when overwritten.
• **Text-speech synthesis:**
The tool uses GTTS to assemble text from subtitles into audio segments, contributing to the overall audio output.[12]

Audio Processing with Spleeter:
• **Separating music with a Spleeter:** The spleeter in the video_to_audio.py module separates music from videos, enhancing the audio experience.

• **Audio-video integration:**
Separated music is added back to the video, enhancing multimedia content with advanced audio.

Web Application for User Interface:
• **Flask Framework:** The delivered Flask web application enables users to interact with the language translation tool using an intuitive interface.
• **User Input Control:** Users can choose which language to go to by streaming videos through a web browser (app.py).
• **Starting the language translation process:** The Start_processing method triggers the processing task, starting the language
translation process.

Real-time updates on Socket.IO:
• **Socket.IO integration:** Socket.IO is used for real-time communication between server and client.
• **New enhancements:** The tool sends new enhancements to the client in the language translation process, and notifies users of
ongoing tasks.

Experiment
1. Set up a virtual Environment: Make sure you have the necessary dependencies installed and create a virtual environment.
2. Run the Flask application: Start the web server and run the Flask application.
3. Upload the video: Use the web interface to upload the video file by clicking the appropriate button. Specify the language in which you want to translate.
4. Processing: The application will start processing the uploaded video. New enhancements will be displayed on the terminal where the Flask application is running. Monitor real-time progress through web interface.
5. Check the results: Once activated, the application should provide a link or button to view/download the video output. Check
the 'uploads' folder for the generated 'output_video.mp4'.
Testing
6. Testing: Test the application with videos and speeches to see how it works in different scenarios.
7. Modify parameters: Experiment with different parameters in code, such as language mapping, resource settings, or translation patterns. Make adjustments as needed and monitor impact.
8. Evaluate results: Check the quality of both decoded audio and video output. Consider things like accuracy, clarity, and consistency.
9. Document findings: List any challenges during implementation and identify possible improvements or expansions of the tool. Document findings, issues, and potential improvements in the report.
10. Shared results: If applicable, share your test results, code changes, and any improvements with the community or in a report.

Spoken Language is Spanish

Spoken Language is converted to English

En este pueblo peruano aún sobreviven y se celebran las tradiciones que trajeron los colonos austroalemanes en el siglo XIX.
In this Peruvian town, the traditions that were brought by Austro-German colonists in the 19th century still survive and are celebrated.

IV. REFERENCES

[1]. VideoDubber: Machine Translation with Speech-Aware Length Control for Video Dubbing Yihan Wu1*, Junliang Guo2 , Xu Tan2 , Chen Zhang3 , Bohan Li3 , Ruihua Song1† , Lei He3 , Sheng Zhao3 , Arul Menezes4 , Jiang Bian2

[2]. Multilingual video dubbing—a technology review and current challenges Dan Bigioi and Peter Corcoran

[3]. MoviePy Library: Newman, Z. (2020). "MoviePy: Video editing with Python." GitHub Repository. https://github.com/Zulko/moviepy

[4]. Spleeter Library: Deezer S.A. (2021). "Spleeter: Deezer source separation library." GitHub Repository. https://github.com/deezer/spleeter

[5]. Whisper Timestamped Model: Whisper Labs. (2022). "Whisper Timestamped Model." Whisper Labs Documentation. https://whisper.ai/docs/whisper timestamped-model

[6]. pyttsx3 Library: "pyttsx3 documentation." Read the Docs. https://pyttsx3.readthedocs.io/en/lat est/

[7]. pydub Library: Jiaaro. (2022). "pydub: Audio processing in Python." GitHub Repository. https://github.com/jiaaro/pydub

[8]. Flask Framework: Grinberg, M. (2022). "Flask: A lightweight WSGI web application framework in Python." Flask Documentation. https://flask.palletsprojects.com/

[9]. Flask-SocketIO: "Flask-SocketIO documentation." Read the Docs. https://flasksocketio.readthedocs.io /en/latest/

[10]. Subprocess Module: "subprocess — Subprocess management." Python Documentation. https://docs.python.org/3/library/su bprocess.html

[11]. ISO Language Codes: "ISO 639-1: Codes for the representation of names of languages." ISO.org. https://www.iso.org/iso-639- language-codes.html

[12]. GTTS (Google Text-to-Speech) Library: Pirate, D. (2022). "gTTS: Python library and CLI tool to interface with Google Text-to Speech API." GitHub Repository. https://github.com/pndurette/gTTS

[13]. Whisper API Documentation: Whisper Labs. (2022). "Whisper API Documentation." Whisper Labs Documentation.

[14]. https://whisper.ai/docs/whisper-api 14. Socket.IO Library: "Socket.IO documentation." Socket.IO. https://socket.io/docs/