Quest Journals
Journal of Software Engineering and Simulation
Volume 9 ~ Issue 4 (2023) pp: 62-75
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

Tic Translation using Attention

Avyay M Casheekar, Nikitha Marium John , Kanishk Rath ,
Dipsikha De , Kaushik S Prabhakar, Sachin A Kishan , Senthil Kumar K
SCOPE, Vellore Institute of Technology University

ABSTRACT : *Tics are involuntary movements or sounds that can occur suddenly and repeatedly, which can make it challenging for individuals with this condition to control their body movements or sounds. For many people, tics are an inborn disability that can lead to difficulties using voice-based applications that require clear and structured speech. This can cause problems with communication and can result in confusion for those who are not familiar with the condition, leading to conversational breakdowns. To address these challenges, the objective is to create a "tic translation" tool that can help people with tics to translate their speech into a format that is easier to understand for those who may have difficulty deciphering what they are saying. This tool could be used to facilitate communication between individuals with tics and those who do not have experience with the condition. The ability to identify and resolve semantic ambiguity is an important problem in natural language processing and is relevant in fields such as information retrieval, question-answering, and chatbots. By developing a tool that can accurately translate the speech of individuals with tics, we believe that we can make significant strides towards improving communication and accessibility for people with tics, as well as advancing the field of natural language processing.*

KEYWORDS: *Natural Language Processing, Transformers, Tourette Syndrome, Tics, Unity, American Sign Language, Flask, Whisper, Open CV*

R*eceived 02 Apr., 2023;* ***R****evised 13 Apr., 2023;* ***A****ccepted 15 Apr., 2023*
P*ublished with open access at* ***www.questjournals.org***

I. INTRODUCTION

Tourette's Syndrome is a neurological disorder characterized by repetitive and involuntary movements or vocalizations known as tics. These tics can be motor tics such as eye blinking, head jerking, or shoulder shrugging, or vocal tics such as grunting, coughing, or even saying inappropriate words. These involuntary tics can cause embarrassment, social isolation, and difficulty in communication with others who are not familiar with the condition.

To alleviate these challenges, this paper proposes a technology-based solution that combines natural language processing (NLP), deep learning, and image processing techniques. This solution is designed to take audio and video input from an individual with Tourette's Syndrome, detect and correct the audio-based tics, and then convert the corrected audio to text. The text output is then displayed on a screen for the person with Tourette's Syndrome to show to their communication partner, thus enhancing their ability to effectively communicate with others.

The proposed application is expected to have a significant impact on the lives of individuals with Tourette's Syndrome, as it will reduce the stigma associated with their condition and improve their ability to communicate effectively with others. Moreover, the technology can be expanded to other application domains, such as semantic ambiguity detection, to enhance communication for individuals with other communication-related disabilities.

*Corresponding Author: Avyay M Casheekar
Scope, Vellore Institute of Technology University

II. LITERATURE REVIEW

Title	**Authors & Year**	**Technique**	**Drawback**	**Remark**
Impact of Tourette Syndrome: A Preliminary Investigation of the Effects of Disclosure on Peer Perceptions and Social Functioning	Brook A. Marcks, Kristoffer S. Berlin, Douglas W. Woods, and W. Hobart Davies **2007**	An attitude change strategy - preventive disclosure has been applied to individuals with TS.	Several limitations should be taken into account - type and severity of tics vary significantly in the participants taken.	The data suggests that preventive disclosure of TS can reduce social rejection and decrease perceptions of drug and alcohol problems.
Public Perception of Tourette Syndrome on YouTube	Mary Jane Lim Fat, Erick Sell, Nick Barrowman, Asif Doja **2012**	First 20 videos under 4 search terms were analyzed. A 3-component rating scale was used, and selected videos were scored by 2 pediatric neurologists using the Medical Video Rating System.	The videos analyzed were limited to the first 20 “hits” for each search term and the first 10 comments.	22% of the videos were negative, and were associated with more negative comments. Negative portrayals were significantly associated with more views.
Attention is all you need	Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin **2017**	Encoder-Decoder architecture with attention mechanisms called Transformer	Generation is highly ordered in nature	Extend the Transformer to problems involving input and output modalities other than text such as audio.
A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling	Yu Wang, Yilin Shen, Hongxia Jin **2018**	Using two correlated bidirectional LSTMs, bi-model based RNN semantic frame parsing network structures are created to accomplish the intent detection and slot filling tasks simultaneously	Solely based on ATIS Dataset, hence it is not trained on naturally spoken data sets and cannot employ more linguistically motivated features.	More parameters need to be updated

		(BLSTM).		
Enhancing Transformer for End-to-end Speech-to-Text Translation	Mattia Antonino Di Gangi, Matteo Negri, Roldano Cattoni, Roberto Dessi, and Marco Turchi. **2019**	SLT adaptation of transformer with integration of ASR solutions for handling long input sequences with low information density.	Encoder side of transformer proves to be problematic	Incorporation of 2d processing of input before self-attention stack and distance penalty in the encoder side
Research on Vocal Tic Symptom Detecting Using SVM/HMM	Su-Seong Chai, InA Kim, and Kyu-Chul Lee **2019**	Used MFCC (speech feature extraction method) and used machine learning algorithms SVM and HMM to generate a recognition model	Bigger the training data was, the more time it takes to educate the recognition model and confirm the output from the input making it very burdensome	Vocal tic detection based on pitch does not solve the core issue of semantic error detection and is very computationally intensive
Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss	Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, Shankar Kumar **2020**	Transformer Transducer model with computer blocks based on self-attention and usage of RNN-T loss well suited to streaming decoding	Is highly computationally intensive and can't be performed locally	The transducer methodology is most efficient at both training and accuracy as compared to LSTMs, RNNs and Vanilla Transformers and can be incorporated into the pipeline.
A deep learning approach for detecting tic disorder using wireless channel information	Arnab Barua Chunxi Dong Xiaodong Yang **2020**	Use CNN model to classify WCI (wireless channel information)- based image data and determine	Requires the use of specialized hardware such as RADAR, USRP etc. to use WCI and detect tics hence making it inaccessible	CNN performs better to classify WCI-based image data as compared to traditional ML models showing scale of model can help with result improvement

Machine Learning for Stuttering Identification: Review, Challenges and Future Directions	Shakeel A. Sheikh, Md Sahidullah, Fabrice Hirsch, Slim Ouni **2021**	Stuttering is classified into two and different ML and DL based approaches such as ANN, SVM, k-NN, Fuzzy Logic, LDA, NBC are applied to find out the best model for Stutter Identification.	Data imbalance issues, or the fact that there are not consistently the same number of samples available for various disfluent categories, affect stuttering datasets as well.	Features applied are MFCCs, LPCC, LFPC, DCT and PP
Tic Detection in Tourette Syndrome Patients Based on Unsupervised Visual Feature Learning	Junya Wu, Tianshu Zhou, Yufan Guo, Yu Tian, Yuting Lou, Hua Ru, Jianhua Feng and Jingsong L **2021**	Uses a DL architecture that combines both unsupervised and supervised learning methods and learns features from videos	The inadequacy of labeled data is a clear limitation to future work	Non-video-based learning methods, such as text based semantic error detection and correction, can be more effective

Table 1 : Literature Survey

III. PROPOSED SYSTEM

3.1 Architecture Diagram

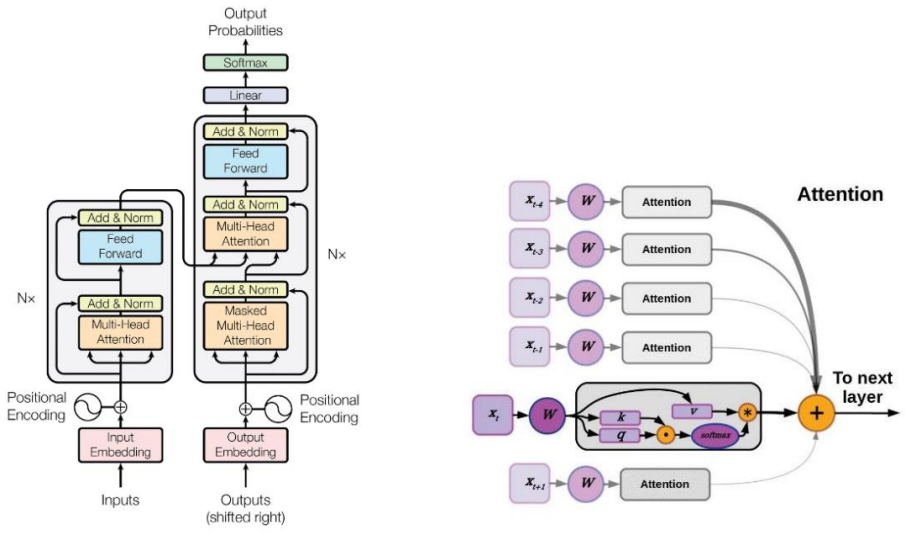

Figure 1: Architecture; Attention Model

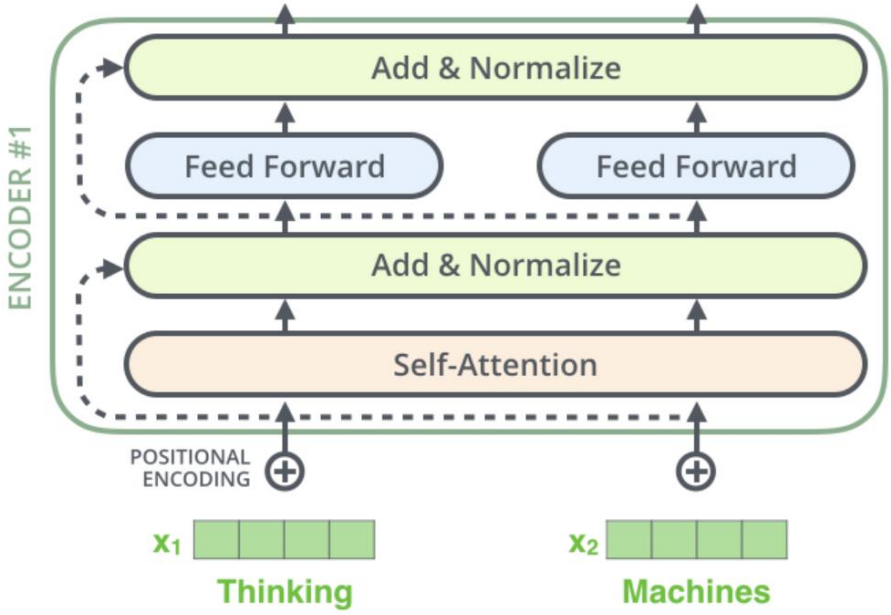

Figure 2: Architecture; Transformer Architecture

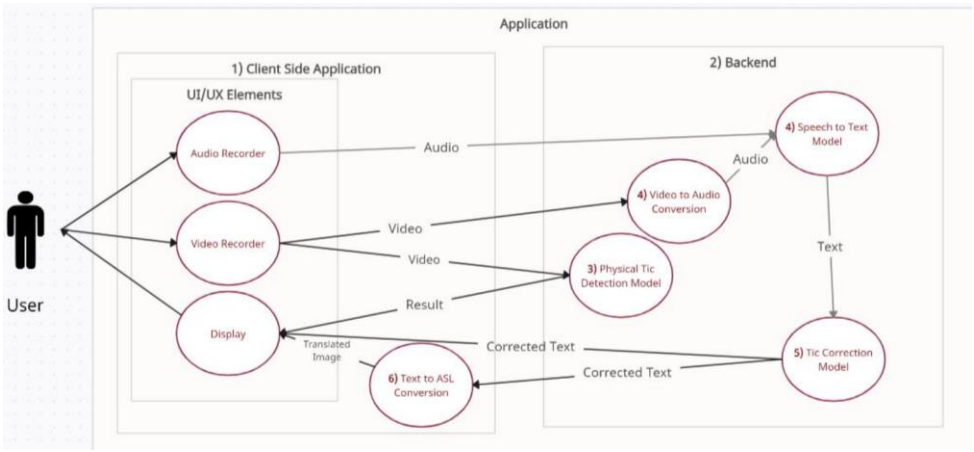

Figure 3: User-Application-Server Interaction and data flow

3.2 UX Design

The front-end application is what the user interacts with to take audio and video recordings. The user can simply press a button to begin recording audio and then presses the same button again to stop recording. For video, the user's camera opens up and records as well. As soon as the recording is complete, the user's recordings are stored in a local location on the phone and are also uploaded to the server.

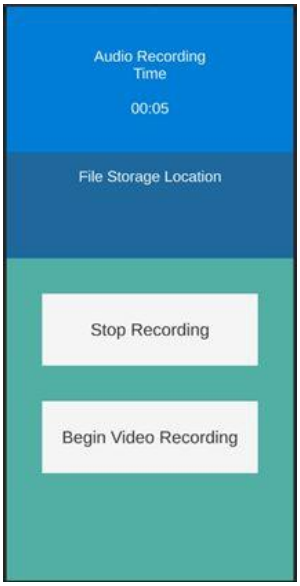

Figure 4: The User Interface of the Application

Audio recording example, where we open the application, press record, and then stop recording to save and send the file to the server. Video works in the same manner

3.3 Server Design

The Back-end of the application consists of a Flask server hosted on PythonAnywhere hosting service, which serves as the middle-ware between the Unity application and the Deep Learning models. Initially, the unity application captures the video or audio of the user which is sent to the server through POST calls made to dedicated routes and stored in the built-in server storage. The server accesses the video/audio data from the server-storage and performs video-based physical tic detection and sends the result back to the unity application where it is displayed for the user. Simultaneously, the server utilizes the deep learning models housed in the server storage to convert the video to audio and then extract the text from the audio which is then supplied as input for the semantic-based sentence tic detection model to identify and eliminate tics in the text extracted from the video. The cleaned text generated by the sentence tic detection model is sent back to the unity application where it is displayed for the user.

3.4 Video Based Tic Detection

An Image Processing model has been developed with the aim of detecting jerky movements that are indicative of tics for the purpose of tic detection. The input video is a prerecorded video that is stored on the server-end. The code commences by importing essential libraries, initializing variables, and configuring the webcam to capture video frames. Subsequently, it analyzes each frame, applies a color-to-grayscale conversion to the image, and subjects it to a Gaussian blur to detect alterations between frames. The absolute difference between the initial frame and the current frame is calculated, and a threshold value of 100 is established to detect abrupt movements. Whenever the absolute difference surpasses the threshold, the system identifies a tic. The system then accentuates the difference between the initial frame and the current frame and detects contours in the frame. Whenever a tic is present, a green rectangle is exhibited around the affected region. The grayscale image, the difference between frames, the threshold image, and the color image with the identified contour are all displayed on the screen. The system halts when the video ends. Whenever a tic is detected, the system displays a message indicating that a tic has been detected; if not, it presents a message specifying that no tics were identified.

3.5 Video preprocessing, Audio to text Conversion

This module runs parallel to the one above. Here, the video is processed suitably to enable audio based tic detection. The processing involves extracting the audio from the input video. The audio is converted into a wav file for easier reading and modifying capabilities. The text is then extracted from the audio file. To achieve this, we make use of Openai's speech to text whisper model. The way Whisper works is by first recording an audio stream, which is then analyzed using a neural network model trained on a large dataset of speech samples. The system uses this model to decode the audio stream into phonemes, which are the basic sound units that make up spoken language. After decoding the phonemes, the model then applies a language model to the phoneme sequence, which helps it to predict the most likely words and phrases that were spoken. The language model takes into account factors such as grammar, syntax, and context to generate the most accurate transcription possible. To ensure maximum accuracy, it also employs advanced techniques such as speaker diarization, which allows it to distinguish between different speakers in a conversation, and noise reduction, which helps to filter out background noise and other audio artifacts that can interfere with transcription accuracy.

3.6 Text Based Tic Detection and Semantic Correction

In this module, we look at an approach for detecting and correcting tics in textual data, specifically focusing on sentences derived from a speech-to-text model. The goal of this process is to identify and eliminate words with low attention values, which are considered as tics, from the input sentences. Once the tics are removed, the corrected text is translated into American Sign Language (ASL) and sent to the front end for further processing.

To achieve this, we employ a Transformer-based model that is capable of detecting the attention values for each token in a given sentence. This model allows us to identify the tics and subsequently remove them from the input text.

The first step in this process is to preprocess the input sentences. This involves the removal of punctuation, tokenization of sentences into words, elimination of numbers, and conversion of words to lowercase. We also considered the removal of stopwords, but this step was not implemented in the final version. After preprocessing, the resulting sentences are then processed by the Transformer model to obtain the attention values for each token.

Next, we extract the words and their corresponding attention values for each sentence in the input. For each sentence, we remove the iterations and retain only the first activation, as we are primarily interested in removing the tics. This results in a list of words and their respective activation values for each input sentence.

To identify the tics, we sort the activation values in ascending order and examine each token, starting with the one with the lowest activation. We consider a token to be a tic if it is a full word present in the input sentence. Once a tic is identified, it is removed from the input sentence. This process is repeated for all sentences in the input text.

Finally, any remaining whitespace in the resulting sentences is eliminated, yielding the corrected text devoid of tics. This text is then used for ASL translation and further processing in the front end.

3.7 Stutter Detection

In this module, we look at an approach for detecting stutters in spoken language by analyzing audio features and utilizing a language model for predicting the next word in a sentence. The primary goal of this method is to identify potential stutters in an audio file and offer suggestions for the next word based on the context, ultimately improving the overall coherence and fluency of the generated text.

The first step in this process involves converting the speech in the audio file into text. To achieve this, we employ an automatic speech recognition (ASR) system, specifically using Google's Speech Recognition service. The input audio file is processed by the ASR system, which generates a text representation of the spoken content.

Next, we calculate two important features from the audio file: zero-crossing rate (ZCR) and energy. These features are essential for detecting speech characteristics, such as intensity and pitch variations, which are commonly associated with stutters. By analyzing these features, we can identify potential stuttering events in the audio file.

Upon calculating the ZCR and energy features, we proceed to detect spikes in both features, which may indicate the presence of stutters. By comparing the spike patterns in both the ZCR and energy features, we can identify overlapping spikes. These overlapping spikes are considered stuttering events, as they correspond to instances where both intensity and pitch variations are observed simultaneously.

With the stuttering events detected, we then leverage a language model to predict the next word in the sentence. In this approach, we use the GPT-2 language model and the corresponding tokenizer for text generation tasks. We construct a prompt for the language model by selecting a few words from the text preceding the first detected stutter. This prompt serves as context for the language model to generate a coherent and contextually appropriate suggestion for the next word.

The GPT-2 model generates a completion for the given prompt, and the last word in the generated completion is extracted as the suggested next word. This word is presented as a possible replacement for the stuttering event in the original text, thus improving the overall fluency of the sentence.

3.8 Sentence to ASL

This module accomplishes the task of converting text to a sequence of corresponding American Sign Language (ASL) images. It works by constructing a dictionary that maps each letter in the vocabulary to its corresponding ASL image filename. The vocabulary includes all lowercase letters from 'a' to 'z', as well as a space character. Once the dictionary is created, the input text is processed by converting it to lowercase and removing any characters that are not in the vocabulary. Then, the text is split into individual characters, which are used to look up the corresponding image filenames in the dictionary. Finally, the image filenames are passed to a function, which accomplishes the task of opening and resizing each image and combining them into a single output image.

The output image is a horizontal sequence of ASL images that represent the input sentence, and is displayed to the user.

IV. EXPERIMENTS AND RESULTS

4.1 UX DESIGN

The Unity Application on the Front-End enables the user to take Video or Audio recordings. When the user starts a new recording, it toggles the camera or the microphone on the device side to record. The user has the option to continue recording for as long as they want. When the user has finished taking the recording, the video/audio file is sent to the Flask API through a POST method. If the upload is successful, the Flask API returns a success message which is shown by the application to the user along with the local file path of the recording.

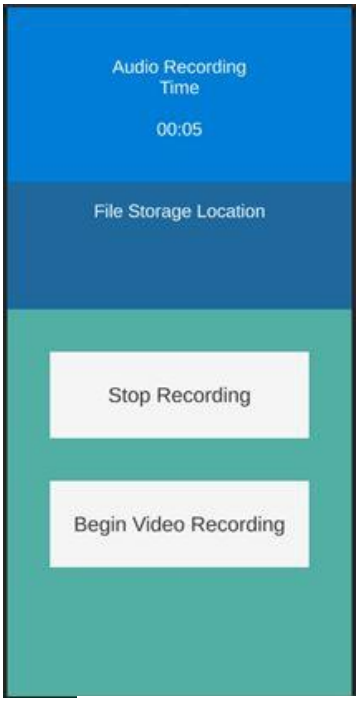

Figure 5: The User Interface of the application

4.2 SERVER DESIGN

The server is configured and managed from the PythonAnywhere server dashboard. The video/audio file is uploaded from the unity application to the Flask API through a POST call and form parameters. When the video/audio upload is successful, it is saved in the server storage as seen below. All the routes and methods are present in the api.py file which is responsible for running the server.

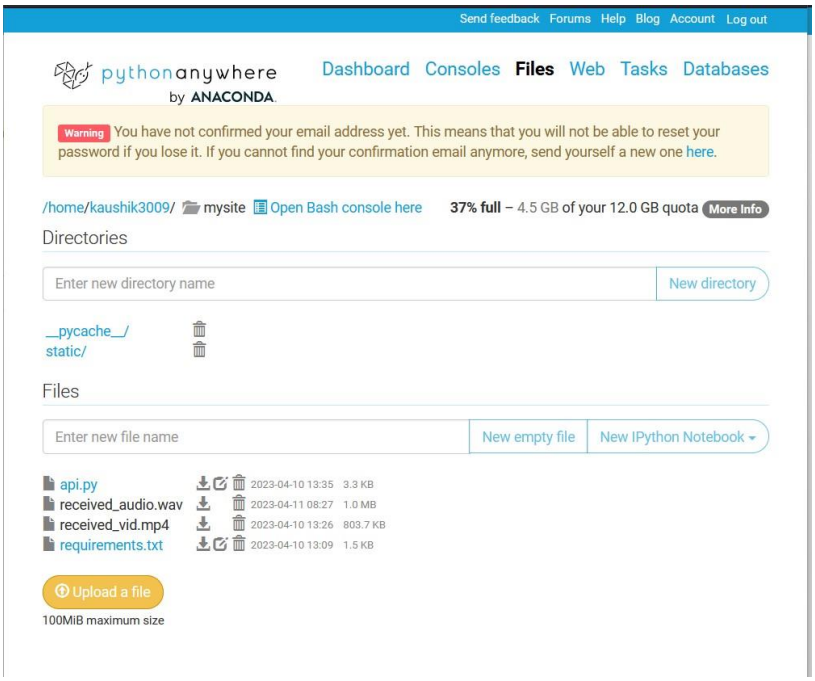

Figure 6: Server Dashboard with Assets

The server functionality for uploading video and audio is tested using PostMan platform. Here, we choose the POST method since we are sending video/audio data to the server. We attach the file in the Body as form-data and run the method. If the file upload is successful, a success message is shown in the output window.

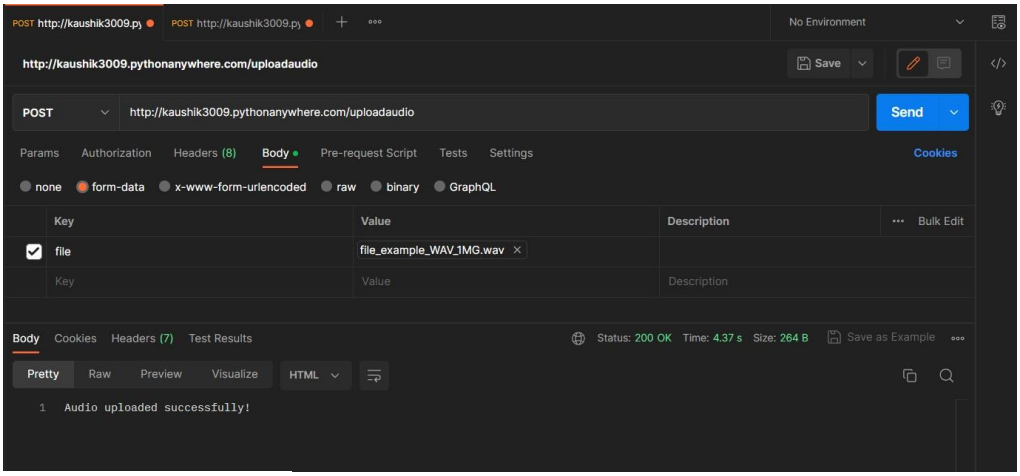

Figure 7: PostMan testing of audio uploads to flask API

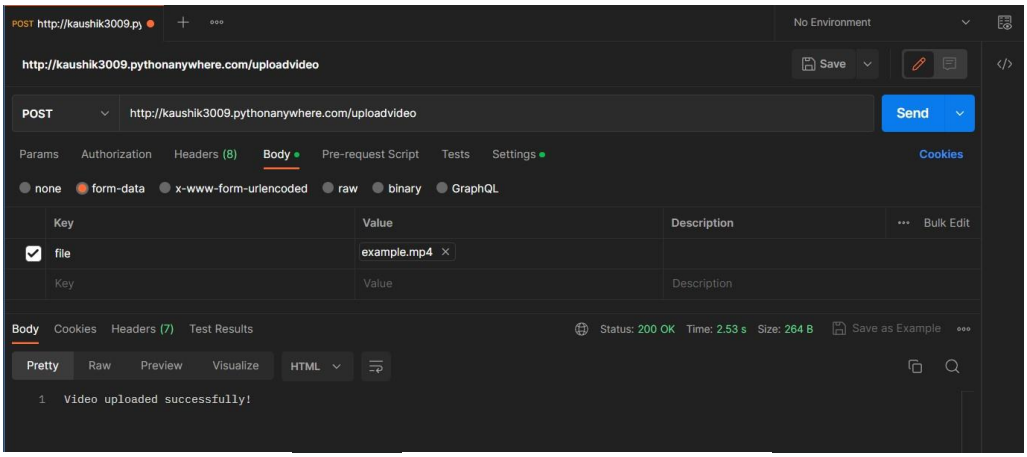

Figure 8: PostMan testing of video uploads to flask API

We perform Video-based Tic detection on the video uploaded from Unity Application. The result of this operation is accessed through a GET method and sent back to the Unity Application where it displayed for the user.

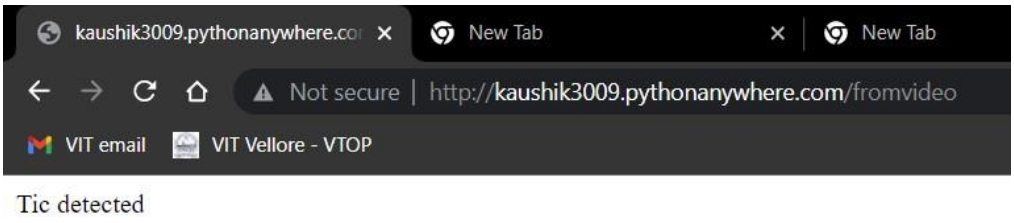

Figure 9: Output of Video based tic detection from server call

4.3 Video Based Tic Detection

A novel Image Processing model has been developed with the primary objective of identifying abrupt movements that signify tics, aimed at detecting tics. The input is a pre-recorded video that is stored on the server-end. Each frame captured from the video is analyzed, converted from color to grayscale, and subjected to a Gaussian blur to detect any alterations between frames. The difference between the initial frame and the current frame is calculated, and a threshold value of 100 is set to identify sudden movements. Whenever the difference exceeds the threshold, the system identifies the presence of a tic. The system accentuates the difference between the initial and current frames, and contours in the frame are

detected. Upon detection of a tic, a green rectangle is displayed around the affected area. The grayscale image, frame difference, threshold image, and color image with the identified contour are all shown on the screen. The system ceases operation when the video finishes. A message indicating the presence or absence of a tic is displayed accordingly

4.4 Video preprocessing, Audio to text Conversion

The video is first loaded in to be processed, following which it the audio is extracted into an mp3 file, and then subsequently into a wav file. The libraries pydub and moviepy.editor are used for this.

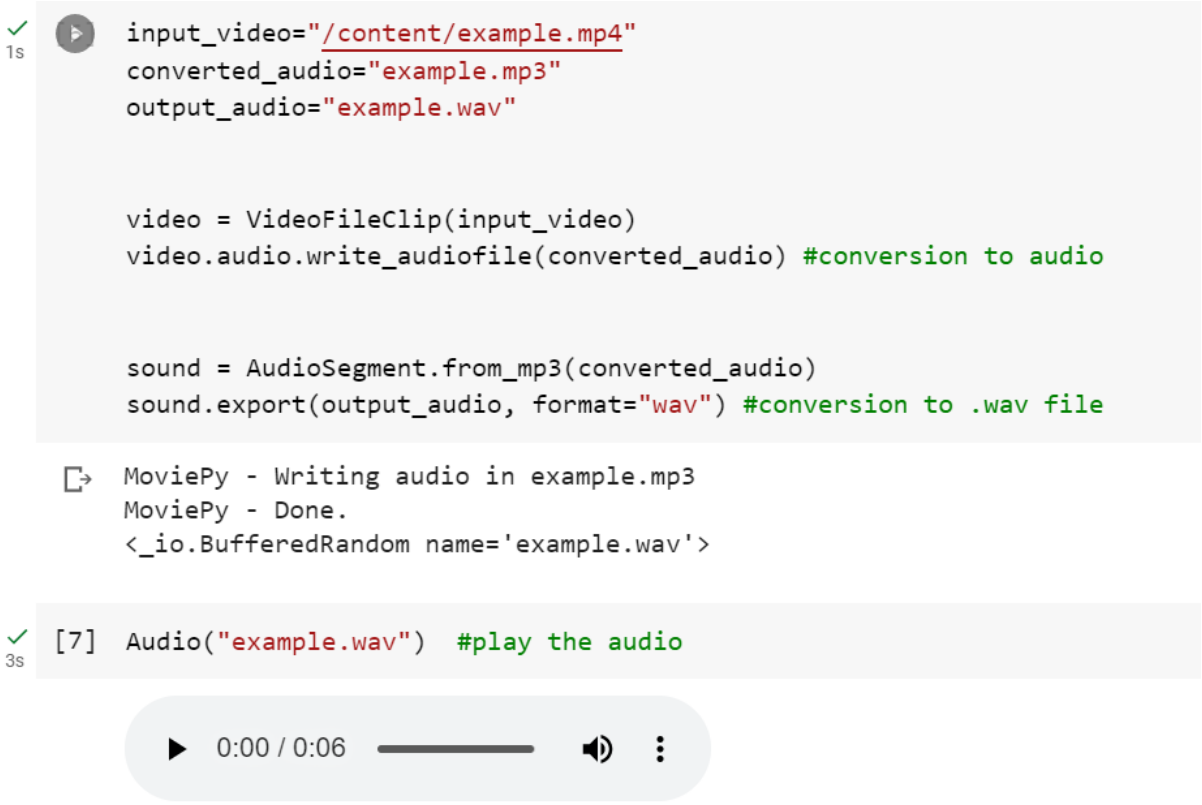

Figure 10: Extracting audio from Video

After successfully extracting the audio from the server, we then load the whisper model. The task for the model is set as transcribe, and the language as English.

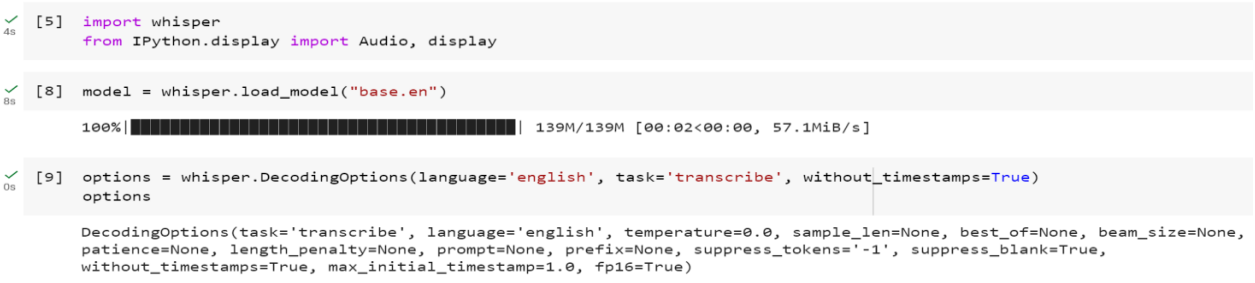

Figure 11: Loading and Initializing the Whisper model

We then process the audio to make it a suitable size, by padding or trimming. This audio is then converted into a log mel spectrogram. This spectrogram is finally passed through the whisper model, where it is processed to give back the recognised speech

```
[10] audio = whisper.load_audio("example.wav")
     audio = whisper.pad_or_trim(audio)

     mel = whisper.log_mel_spectrogram(audio).to(model.device)
     result = model.decode(mel, options)
```

```
result.text
```

```
'Hello, good morning to you all. I hope you're having a good day.'
```

Figure 12: Passing the processed Audio through the model

4.5 Text Based Tic Detection and Semantic Correction

The script presented in this study preprocesses and simplifies input sentences by identifying and removing words with low activation values in a given neuron. We demonstrate the functionality and output of the script using three example sentences:

t e x t l i s t = [
" I hate bald s c h o o l " ,
" I own a cr ap s t o r e " ,
" I e a t c h a i r f o o d "
]

These sentences can be replaced with any custom sentences for further exploration. The script preprocesses the input sentences by removing punctuation, converting words to lower-case, and tokenizing them. The preprocessed sentences are then visualized, as shown below:

t e x t l i s t = [
' i hate bald s c h o o l ' ,
' i own a crap s t o r e ' ,
' i e a t c h a i r food '
]

The Neuron Text Simplifier is applied to the preprocessed sentences, iteratively calculating activations and removing the word with the lowest activation value. The final simplified sentences are printed, and a visualization of the simplification process is generated:

Figure 13: Simplified Text Visualization and Attention coloring

The output consists of the original sentences with the identified tics removed. In our example, the following simplified sentences are obtained:

t e x t l i s t = [
P r e p r o c e s s e d Text : [
' i hate bald s c h o o l ' ,

```
' i own a crap s t o r e ' ,
' i e a t c h a i r food '
]
S i m p l i f i e d Text : [
' i hat e s c h o o l ' ,
' i own a s t o r e ' ,
' i e a t food '
]
]
```

Users can adjust the input sentences, model, layer, or neuron parameters to observe how different configurations affect the script's performance. By exploring these variations, this research aims to provide insights into the process of text simplification and contribute to the development of more effective natural language processing tools.

4.6 Stutter Detection

The presented script aims to detect stutters in an audio file and suggest the next word using a pre-trained GPT-2 language model. It employs both speech recognition and transformer-based text completion techniques to accomplish this task. The audio file is first transcribed into text using Google Speech Recognition. Next, the zero-crossing rate (ZCR) and energy features are calculated, and the spikes in these features are detected. Stuttering events are identified by finding overlapping spikes in both the ZCR and energy features. If stuttering events are detected, the script uses the last few words before the first stutter as a prompt to suggest the next word based on the GPT-2 language model. The number of words in the prompt can be adjusted to provide a better context for the language model:

```
prompt = " " . j o i n ( words [ − 5 : ] )
```

Finally, the script outputs the suggested next word or a message indicating that no stutters were detected. The output for the example audio file is as follows:

```
S u g g e s t e d next word: [ INSERT SUGGESTED WORD HERE]
```

By adjusting the input audio file and the number of words in the prompt, users can explore the performance of the script under various conditions. This research aims to provide insights into stutter detection and contribute to the development of more effective speech processing tools.

4.7 Sentence to ASL

The script converts a sentence into a sequence of American Sign Language (ASL) images by matching each letter in the input sentence with its corresponding ASL image file, using a pre-defined vocabulary of letters. The resulting sequence of image files represents the input sentence in ASL. These image files are then merged horizontally into a single image, which is displayed using the Python Imaging Library (PIL). The output for the sentence" How are you?" is given below:

Figure 14: Sample Output Image

V. CONCLUSION

In conclusion, this study has presented a pioneering approach towards addressing the communication challenges faced by individuals with tics, an involuntary condition that affects body movements and sounds. Through the development of a" tic translation" tool, we have taken a significant step in facilitating better communication between people with tics and those who may be unfamiliar with the condition. This tool holds great potential in improving the accessibility and inclusivity of voice-based applications for individuals with tics, making these platforms more accommodating to their unique needs. The" tic translation" tool is not only beneficial for individuals with tics but also has broader implications for the field of natural language processing. By working on semantic ambiguity resolution and enhancing the tool's translation capabilities, we are contributing to the advancement of natural language processing techniques. This, in turn, can positively impact various applications, such as information retrieval, question-answering, and chatbots. Furthermore, the development of this tool highlights the importance of considering the diverse needs of different user groups when designing and developing technology. By focusing on the challenges faced by individuals with tics, we

emphasize the need for inclusive design principles, ensuring that technology can be accessible and usable by a wide range of users.

This study serves as an essential reminder of the potential for technology to bridge gaps, foster understanding, and promote inclusivity in society. The" tic translation" tool can serve as a foundation for future work that seeks to refine, expand, and enhance its capabilities. By continuing to explore the needs of individuals with tics and incorporating their feedback, we can ensure that our approach remains user-centered and continually evolves to better address their communication challenges.

VI. Future Work

In future work, it will be essential to further refine the" tic translation" tool by incorporating user feedback and exploring additional features, such as real-time translation and integration with popular communication platforms. Expanding the tool to support multiple languages and cultural contexts will also be crucial in making it more universally accessible. Additionally, investigating the potential application of advanced natural language processing techniques, such as transformers and deep learning models, could improve the tool's translation accuracy and adaptability, ultimately enhancing its overall effectiveness in facilitating communication for individuals with tics.

REFERENCES

[1]. Fat, M. J. L., Sell, E., Barrowman, N., & Doja, A. (2012). Public perception of Tourette syndrome on YouTube. Journal of Child Neurology, 27(8), 1011-1016.

[2]. Marcks, B. A., Berlin, K. S., Woods, D. W., & Davies, W. H. (2007). Impact of Tourette Syndrome: a preliminary investigation of the effects of disclosure on peer perceptions and social functioning. Psychiatry, 70(1), 59-67.

[3]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polo-sukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[4]. Di Gangi, M. A., Negri, M., Cattoni, R., Dessi, R., & Turchi, M. (2019, August). En-hancing transformer for end-to-end speech-to-text translation. In Proceedings of Machine Translation Summit XVII: Research Track (pp. 21-31).

[5]. Zhang, Q., Lu, H., Sak, H., Tripathi, A., McDermott, E., Koo, S., Kumar, S. (2020, May). Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 7829-7833). IEEE.

[6]. Wang, Y., Shen, Y., & Jin, H. (2018). A bi-model based rnn semantic frame parsing model for intent detection and slot filling. arXiv preprint arXiv:1812.10235

[7]. Sheikh, S. A., Sahidullah, M., Hirsch, F., & Ouni, S. (2022). Machine learning for stuttering identification: Review, challenges and future directions. Neurocomputing.

[8]. Chai, S. S., Kim, I., & Lee, K. C. (2017). Research on Vocal Tic Symptom Detecting Using SVM/HMM. Advanced Science Letters, 23(10), 9694-9697.

[9]. Barua, A., Dong, C., Yang, X. (2021). A deep learning approach for detecting tic disorder using wireless channel information. Transactions on Emerging Telecommunications Technologies, 32(7), e3964.

[10]. Wu, J., Zhou, T., Guo, Y., Tian, Y., Lou, Y., Ru, H., ... Li, J. (2021). Tic detection in tourette syndrome patients based on unsupervised visual feature learning. Journal of Healthcare Engineering, 2021