AutoNote

Jason Fagerberg, Eric Wahlstrom, Calvin Phung, Kevin Mick and Mira Yun
Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, MA 02115, USA
{fagerbergj1, wahlstrome, phungc, mickk, yunm}@wit.edu

***Abstract*—Online courses and mobile applications have been offered to provide diverse and convenient learning opportunities. However existing online learning services are not designed to help audio and visual processing disorders. *AutoNote* is a web-based platform that provides audio and white board transcriptions from online learning services offered with any video presentations. By using Watson's audio recognition service and machine-learning technology, *AutoNote* provides both audio and white board transcriptions from videos where the users can explore and learn at their own pace.**

Keywords*—*Transcription Generator, Machine Learning, Image Processing, Audio Recognition.

I. INTRODUCTION

In a classroom or meeting setting, it can be difficult to keep up to speed with what the lecturer or presenter is presenting. People often struggle to determine what to write down during lectures or meetings. There is a lot going on during a lecture or presentation, so the individual may not be able to capture every main concept perfectly. For some people, it could be a slow processing speed issue or the lecturer's rapid presentation skills.

Processing speed is the pace at which an individual take in information, make sense of it, and begin to take another course of action. This information can be visual, such as letters and numbers, or auditory, such as spoken language [1]. Having slow processing speed has nothing to do how smart an individual is - just how fast they can take in and use information. For example, when a student with slow processing speed is in class taking notes and he/she sees a word that they are not familiar with, the student will have to slow down to understand the meaning of the word, which makes it more difficult for the student to keep up with the lecturer [1].

Hearing problems such as auditory processing disorder (APD) is deemed as issue as well when it comes to learning. When an individual has APD, they are unable to process the information they hear due to miscommunication between the ears and brain. They have difficulty recognizing subtle differences between sounds in spoken words. This is very common within a typical classroom when there is background noise.

The most common learning disability is dyslexia. Dyslexia is a problem with language, not vision. It causes difficulty with reading. A key sign of dyslexia would be troubling decoding words. In a classroom or meeting setting, the individual would miss out many of the important details due to these conditions [2]. Accommodations that could help with Dyslexia include the following: extra time for reading and writing, extra time on exams, allowing the individual to show comprehension in different way, and simplified directions [3].

Current mobile phone applications such as Lumosity [4] and Elevate [5] are designed to help train the brain to build a daily habit around acquiring how an individual thinks. These applications put critical thinking, memory, and problem-solving skills to the test. Online courses such as Lynda by Linkedin [6], Coursera, Udemy, edX, and Khan Academy are also alternative solutions for those who has visually or auditory processing issues. Lynda is an American massive open online course website that offers video courses taught by industry experts in software, creative, and business skills. They generate learning paths that are comprised of multiple videos pertaining to the specific subject matter. Each video that's watched, Lynda offers sections with available timing and transcripts, as shown in Figure 1.

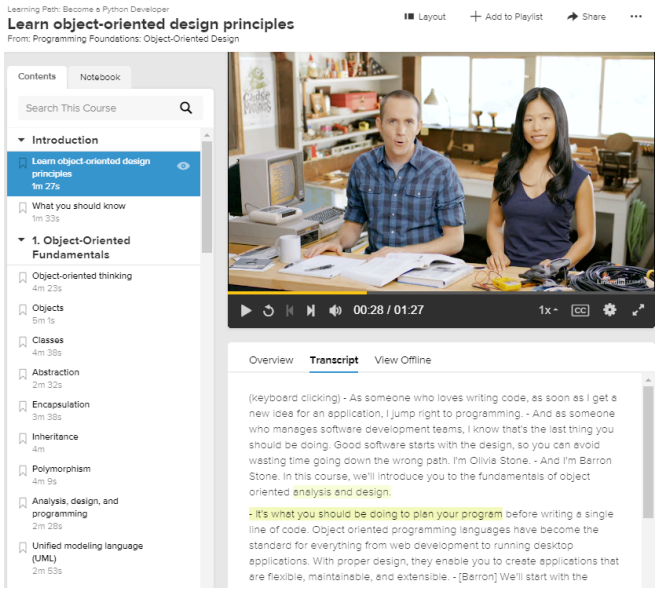

Fig. 1. Lynda's Interface

Online courses such as Lynda are very beneficial towards those who cannot afford a college education and would like to only obtain certificates within a certain subject matter. This does not solve the issue of learning within a classroom or meeting without missing crucial details. Massive open online courses (MOOCs) benefit those interested in cyber learning, but it may not always be applicable to the subject matter a person is looking to learn. With problems such as Audio/Visual processing disorders, *AutoNote* is created with the goal of

improving the lifestyle of those who have trouble processing important information in a classroom or work setting.

The remainder of this paper has the following sections. Section II presents how each module of *AutoNote* was designed and implemented. Section III concludes our work and addresses the possible improvements that can be made to the system in the future.

II. AUTONOTE

AutoNote is a web-based platform that provides audio and white board transcriptions from online learning services offered with any video presentations. From the video presentation, *AutoNote* extracts audio transcription through audio processing algorithm and Watson's audio recognition service. In addition, *AutoNote* creates a board transcription through a machine-learning model from the word and letter image extractions of the video frames. These transcriptions will be passed back to the front end where the user can explore the transcriptions connected to their video and learn at their own pace.

2.1 Front End

The front end website is a single page application that was built using the reactJS framework for function and bootstrap4 for styling. ReactJS utilized a domain object model to dynamically change elements in a single HTML file. These elements may be contained in variables or objects known as components. The base for the website is the div that is styled with the bootstrap root tag and colored with custom CSS. The main controller for swapping UI elements and components will be the applications state variable. This variable can have a value of PENDING, PROCESSING, or RESULT.

The initial state for the application is PENDING. In this state, the welcome component will be rendered. The welcome component is a bootstrap jumbotron, which contains some welcome and instructional text and an upload video button as shown in Figure 2. When the user clicks the upload video button, a standard video file input dialog will appear. The user will only be allowed to select video files in this dialog. After the user selects a video file, the file will be placed into a form data object created using the standard form data library [7]. The form data object will then be passed to a processPost() function along with arrow functions that handle the data after a response is received and that displays error messages if the request failed.

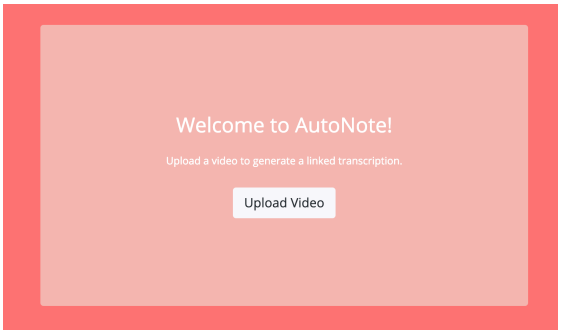

Fig. 2. Welcome Component

In the PROCESSING state of the application, the upload button will be replaced with a loading spinner and a cancel button. The application will remain in this state until a response is received from the API. If the user presses the cancel button, then a new request is sent to the API with a null file. If the post was successful, then the callback() function will be called. This callback() function will create a URL from the video file using the standard URL library [8]. This function will also parse the *boardTranscription* and *audioTranscription* lists from the response. After this data is saved, the state of the application is changed to RESULT.

In the RESULT state, the welcome component is swapped for the result component as shown in Figure 3. The result component contains a back button, the video uploaded by the user, two tabs for switching transcriptions, a text box that holds the transcription, and a footer that contains a download results button.

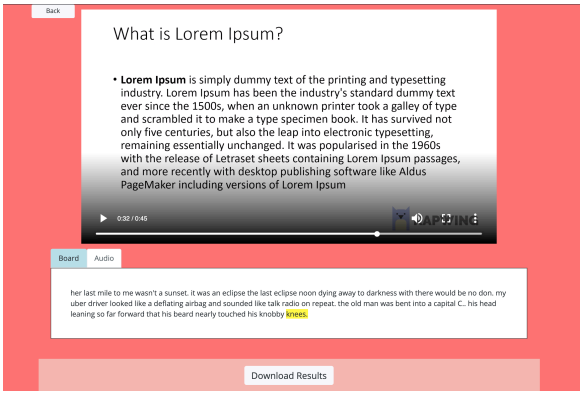

Fig. 3. Result Component

Every quarter second, the application will loop through the active transcription looking for the line that has a time stamp that is closest to the current video's time without going over. Once the closest line is found, it is then selected and highlighted. This allows the application to always have the most relevant transcription line highlighted yellow so the user can track the transcription as it is said or written. Since this function is run so often, some measures were taken to help increase the time efficiency. If the current video time is equal to the currently selected transcription timestamp, the function returns. If the current video time is greater than the current transcription line timestamp then each of the next transcriptions lines is checked until the correct line is found.

The user may hover over any line in the transcription and see the corresponding timestamp as shown in Figure 4. The user may also click on this line to skip the video to that corresponding timestamp.

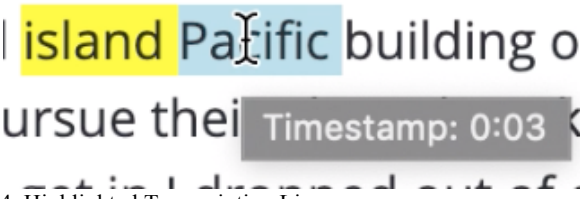

Fig. 4. Highlighted Transcription Line

The download results button is an HTML link element that downloads the files from a URL that is created when the link is clicked. The text from the transcriptions is parsed out of the transcription lines and saved to a zip file by utilizing the JSZip library [9]. Then the standard URL library creates a URL that references that zip file so the user can download it when they click the link [10].

2.2. Image Processing

For the transcription process to take place, written information needs to be gathered from the submitted video. The video goes through a process that strips individual frames and collects their timestamps to be referenced post process. By taking the framerate of the video, the program can compute the correct time of each frame.

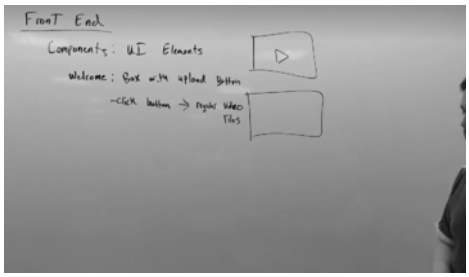

Fig. 5. Gray scaled video frame

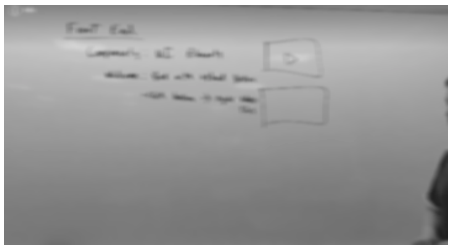

Fig. 6. Blurred video frame

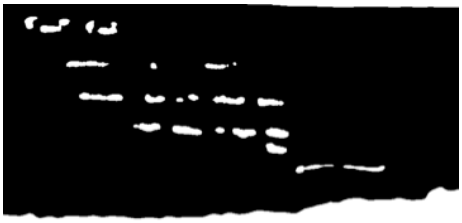

Fig. 7. Video frame with Black-white threshold filter

The individual frames go through a process of image manipulation that will allow the program to recognize words and letters, based on the contours of each written element. The manipulation to get the words from each frame is a three-step process that takes each frame through a series of changes [11]: Grady-scaling (Figure 5), Gaussian Blur (Figure 6), and Black-white threshold (Figure 7).

Through OpenCV2's function “findContours”, the program approximates boundaries for the words that will allow for the program to be able to “group” each individual word based on the contour of the blurred words, these worlds will be saved as separate images. In Figure 7, each word can be seen as a white blur, through OpenCV2 the boundaries of the white “words” will be used by the program to collect the area of each element to be split. A separate directory will be created to collect every word from the frame that was processes.

Fig. 8. Letter stripping of a word

Now that the words written on each frame are collected, the next step is to split each word into their individual letters so they can be analyzed by the CNN. A similar process takes place to strip the letters from the words. First, the images of the previously processed words are taken and processed in a similar fashion as before, except now, instead of blurring the images to group them the blurring process is skipped to retain accurate contours of each letter: Gray-scaling and Black-white threshold. Skipping the blurring step results in accurate boundaries for each letter, Figure 8 displays the boundaries of each letter to be stripped from the word, which can now be sent to the letter recognition model to give a prediction on what letter was found.

2.3 Letter Recognition

We have trained a model by utilizing the EMNIST [12] data set, which is comprised of over 800,000 images of hand written alphanumeric characters. We use this data set with a convolution neural network to build a model that can predict hand written letters, and the Tesseract-OCR library tricking eyes images that are potentially not alphanumeric characters, that our Computer vision software picks up.

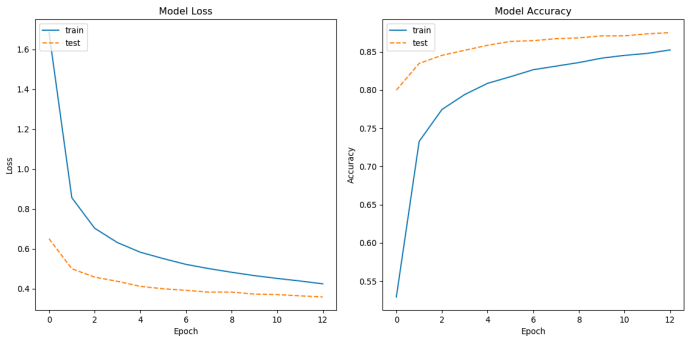

Fig. 9. Our current models level of accuracy (blue) compared of the test data (orange).

To briefly elaborate on what a convolution neural network, it is a machine-learning algorithm that is mainly used for image recognition [11]. The optical character recognition (OCR) Tesseract program is generally used for converting printed texts such as alphanumeric characters on credit cards, billboards, or in textbooks into digital files. Unfortunately however, it is not trained for handwritten letters and therefore works very poorly with our images. For our project however, when we check a character against our model it returns the results array from Figure 9,which is a list of probability between zero and one we check to see if the highest probability is less than 75% if so we can utilize the Tesseract library due to the likelihood that what the image is is more likely a symbol such as a period, comma, or some other possible symbol.

2.4 Audio Recognition

Speech recognition is the ability of a program to identify words and phrases in spoken language and convert them to a machine-readable format. Rudimentary speech recognition is deemed imprecise at times and may only identify words if they are spoken very clearly and slowly. Algorithms are utilized through acoustic and language modeling. There are many available packages for speech recognition out there that works with *AutoNote*, such as google-cloud-speech, assemblyai, apiai, pocketsphinx, and Watson-developer-cloud. *AutoNote* takes advantage of IBM Watson's Speech to Text APIs to produce transcripts of spoken audio [13]. *AutoNote* utilizes IBM Watson's Speech to Text synchronous and asynchronous HTTP REST interfaces. Users are able to send requests and receive results over a single connection asynchronously, thus resulting in faster transcription. Figure 10 shows the process of implementing IBM Watson's Speech to Text API libraries and receiving the transcribed audio text.

```
def get_audio_transcription(self, video_file):
        stt =
SpeechToTextV1(    iam_apikey='t0rpKCce8HfPqMi
e-2bz3QPi301_0Dj5Nfc-ypuxsb1m',
url='https://gateway-
wdc.watsonplatform.net/speech-to-text/api')
        clip = mp.VideoFileClip(video_file)

clip.audio.write_audiofile("theaudio.mp3")
        audio_file = open("theaudio.mp3",
"rb")

        with open('transcript_result.json',
'w') as fp:
            result = stt.recognize(audio_file,
content_type="audio/mp3",

continuous=True, timestamps=True,

max_alternatives=1)
```

Fig. 10. Code snippet of audio transcription

III. CONCLUSION AND FUTURE WORK

AutoNote aims to provide an on demand and comprehensive experience that modern online learning sites provide, but with any lecture video. *AutoNote* generates audio and white board transcriptions from online learning services offered with any video presentations. From the video presentation, *AutoNote* extracts audio transcription through audio processing algorithm and Watson's audio recognition service. From the video frames, *AutoNote* extracts a white board transcription through a machine-learning model of image processing. These transcriptions will be passed back to the front end where the user can explore the transcriptions connected to their video and learn at their own pace.

AutoNote is our very first solution for extracting transcriptions from online learning videos. The path that word and image files are created does not allow multiple users to access a server at a time, in the future this should be changed to allow as many users to process videos at the same time as possible. The processing speed of the server program can be improved by multithreading the letter detection portion of the algorithm as well as making the audio recognition portion it's own thread.

REFERENCES

[1] G. Charbonneau, A. Nertone, F. Lepore, M. Nassim, M. Lassonde, L. Mottron, O. Collignon, "Multilevel alterations in the processing of audio–visual emotion expressions in autism spectrum disorders," Neuropsychologia, 2013.

[2] U. Team, "Understanding Dyslexia," Understood.org. https://www.understood.org/en/learning-attention-issues/child-learning-disabilities/dyslexia/understanding-dyslexia.

[3] P. Rosen, "The Difference Between Dyslexia and Dyscalculia," Understood.org. https://www.understood.org/en/learning-attention-issues/child-learning-disabilities/dyslexia/the-difference-between-dyslexia-and-dyscalculia.

[4] https://www.lumosity.com

[5] https://www.elevateapp.com/

[6] https://www.lynda.com/

[7] Form-Data, "form-data/form-data," GitHub, 17-Oct-2018. https://github.com/form-data/form-data#readme.

[8] Axios, "axios/axios," GitHub, 04-Mar-2019. https://github.com/axios/axios.

[9] JSZip. https://stuk.github.io/jszip/. \

[10] K. Kelly, "Processing Speed: What You Need to Know," Understood.org. https://www.understood.org/en/learning-attention-issues/child-learning-disabilities/information-processing-issues/processing-speed-what-you-need-to-know.

[11] Githubharald, "githubharald/WordSegmentation," GitHub, 26-Aug-2018. https://github.com/githubharald/WordSegmentation.

[12] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," arXiv.org, 01-Mar-2017. https://arxiv.org/abs/1702.05373.

[13] A. Nadas, "Estimation of probabilities in the language model of the IBM speech recognition system," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 4, pp. 859-861, August 1984