Abstract – The proliferation of new AI technologies based on deep learning and powered by massive industrial data sets has enabled the widespread dissemination of easy-to-use, cloud-based, artificial intelligence APIs for application development. Instead of expanding upon a specific facet of foundational research, the goal of this project is to explore the utility of some of these APIs, and to develop new and valuable usage scenarios for these AI-based utilities. Specifically, this project focused on the development of a cross-platform text-to-speech (TTS) system that leveraged cutting-edge optical character recognition (OCR) and human-like speech synthesis endpoints to extract text from captured images and read that text aloud to users. In time, hopefully this application and applications like it will serve as valuable learning tools for the visually impaired and dyslexic.

Index Terms – Optical Character Recognition, Human-Like Speech Synthesis, Dyslexia, Disability.

INTRODUCTION

In the United States today, somewhere around 5-10% of school children suffer from dyslexia¹, and roughly 3% are either completely blind or in some other way severely visually impaired². To put that in numbers, between 26 and 42 million Americans are faced with reading based impairments caused by these common disabilities alone. At the same time, nearly 77% of the American population owns a smartphone equipped with a camera³, and even simple OCR systems are performant enough to succeed at basic reading tasks.

Also, the growth and development of virtual assistants such as Amazon’s Alexa and the Google Assistant have sparked increased interest and development of human-like generative speech systems. These platforms hope to leverage generative speech systems to enable their speakers to interact with the world using dynamic, realistic spoken language. However, these systems can be leveraged for much more.

Finally, the recent movement toward cloud computing has made it such that the “edge” computing environment that data is acquired from and presented on has no significant effect
on the type of computation that a given application can carry out. Whether an individual is interfacing with a website, a mobile application, or a voice assistant, it’s safe to assume that any serious computation will be carried out on a high throughput remote server somewhere off in the cloud (i.e. in some giant data center attached to the edge device via the internet).

These trends inspired the creation of the cross-platform, computer vision based text-to-speech application detailed in this paper.

**BACKGROUND**

While building and deploying handwritten OCR or humanlike text-to-speech endpoints can be prohibitively difficult and expensive, recent industry developments have made these AI components extremely accessible. Unfortunately, using corporate AI APIs has made it difficult to explicitly define the specific processes that power their AI utilities. While the Microsoft Azure Computer Vision API (upon which the OCR for this application relies) parallels projects such as the Google-backed Tesseract, and AWS Polly, which powers the app’s text to speech engine, parallels Google’s Tacotron, the specifics of the implementations seen in the current app are unfortunately opaque.

In the future, it would be interesting to continue exploring the use of open-source software as the core engines of the application. However, cursory research and experimentation indicated lower overall parse accuracy with Tesseract, and prohibitively slow voice generation with Tacotron. In both cases, these systems were not competitively optimized, and they were not run on comparable servers to the AWS or Azure backend that supports the corporate endpoints.

The framework upon which this application was built is Xamarin.Forms. Xamarin.Forms was chosen for a few reasons. First, as a cross-platform framework, it widens the scope of the overall addressable set of users to anyone with an iOS, Android, or Windows device. Given that 85% of desktop PCs run some version of Windows, and that within 85% of the households that support then 11% macOS market share, a Windows PC is also present, and acknowledging that that .5% of ChromeOS users will soon be able to run Android applications on their device, this seemed like a reasonable decision.

Second, Xamarin.Forms is based in the C# programming language. C# is the language of choice for Microsoft-derived applications, and thanks to their relative corporate dominance, great resources exist for debugging and developing C# applications, especially ones which
interface with the corporate AI resources that were used in the development of this application.

Third, Xamarin.Forms provided a series of valuable hooks into the host operating system for Android, iOS and Windows devices, while still making it simple and accessible to write platform-specific code using compiler directives. Given that this application needs to interact with the host operating system occasionally in uncommon ways, this feature provided a great deal of flexibility while developing the application. Moving forward, this app should remain viable as a foundation for future development, even if this development requires extensive platform-specific code.

Beyond the core user numbers, the importance and significance of an AI-driven text-to-speech platform like this is validated by platforms such as the Learning Ally (previously known as Reading for the Blind and Disabled) audiobook library. Learning Ally’s audiobook library is currently the largest human-narrated audiobook library in the world\(^8\), and a majority of its content is academic material for elementary through high school students. While its existence is commendable, access to the library costs $119 per year, which is prohibitive to many, and due to its volunteer-driven nature, the quality of narrated audio is highly inconsistent.

Additionally, Learning Ally is primarily focused on enabling younger students to attain a standard education, and its utility drops off severely as users progress through school. Given that dyslexia and visual impairments are most commonly carried throughout life, this poses a significant problem. Finally, it’s important to note that the volunteer-reader based approach utilized by learning ally makes it virtually impossible for them to record and distribute audio of brand-new publications or cutting-edge research. While many new publications ship with audiobook counterparts, purchasing professional audiobooks en masse can often be cost-prohibitive.

**BASIC OCR/TTS PIPELINE**

The basic processing pipeline employed by the application can be broken down as such:

1) From the main view of the application, a user either captures an image, or (using the app) browses through their device’s file system to select a series of images from which they would like to hear spoken text.

2) That image or set of images are decomposed into a list of C# byte arrays, which are sent up to the Azure Computer Vision OCR endpoint.
3) From the Azure Computer Vision OCR endpoint, a list of strings of parsed text is received. This list of all extracted strings is passed on to the next view for processing, along with flags indicating whether the user desires to use their OS’s basic TTS system or Amazon Polly, and whether they would like the text to be “cleaned up” using an effective but slightly time-intensive spellchecking process.

4) Upon receiving the list of extracted strings, the backend for the new view immediately re-processes the array of parsed strings so that strings break on sentence delimiting characters such as periods or question marks instead of the random breaks determined by the API. This helps TTS systems “understand” the flow of a sentence, and pronounce sentences more naturally.

5) Should the spellchecking flag be set, each line is sent to the Azure Spell Check API for correction, with all preceding and following lines of text also given to the API for context. As the spellchecking routine iterates through each line of text, it updates that line with corrections, and those corrections are then leveraged to improve the quality of the “preceding text” fed into future calls. At the moment, this process is run only once (for time efficiency), so later strings often see a higher quality parse than earlier ones. In the future, it might be interesting to explore what number of iterations through this process produces the highest quality parse within a reasonable span of time.

6) Once text has been re-delimited and (optionally) spellchecked, its given over to a TTS engine. In the case of Amazon Polly, this TTS engine works by defining each line of text as a distinct object, calculating an MP3 file that represents the text spoken aloud, saving that MP3 file to a temporary location in local storage on the device, and linking that temporary MP3 file to the object in such a way that these objects can be iterated through by a given system’s media player. Should the user instead prefer to use a native TTS engine, the array of processed strings is just dynamically partitioned in response to user interaction, and the entirety of the generated text partition is fed into the TTS engine in response to a user request.

7) Once all of this data has been processed, a basic UI page is rendered showing each line of processed text (which can be clicked on to begin playing at that position), along with basic audio playback controls. For the AWS Polly implementation,
playback of each processed line of text can either be controlled with these controls, or via the system’s native audio playback interfaces.

**FUTURE DEVELOPMENTS**

There are a handful of valuable features that could be built into this platform to increase its utility, especially amongst disabled populations.

1) Now, a list of images can be parsed, processed and read aloud sequentially. However, there exists no in-app method to disassemble a multi-page PDF document and read it aloud page-by-page. This could be solved by either hosting a server that could receive multi-page PDFs, split them (using a program such as ImageMagick or ghostscript) and then return a list of PNG files, or by leveraging an already existent API such as the CloudConvert PDF \(\rightarrow\) PNG API. This would be valuable for the visually impaired, as it would require less precise user interaction.

2) It might be very valuable to make the extracted text that is read aloud searchable. Especially given the large comorbidity of dyslexia and attention deficit disorder, building out features to help users who have lost track of the text being read aloud could be extremely useful. In a similar vein, right now the system for highlighting what text is being read aloud is imperfect and buggy. Improving this could help users track where they are in a page that is being read to them.

**CONCLUDING REMARKS**

Even in its current, very prototypical state, this platform highlights the immense ability of current AI knowledge to be leveraged to help disabled populations. While hosting and deploying hand-written systems for OCR and humanlike voice synthesis is currently quite difficult and expensive, there exist many powerful and performant AI APIs provided by major companies that enable everyday programmers to build AI-based systems, and as time moves forward, these APIs will only become faster and more powerful.

**REFERENCES**


AUTHOR INFORMATION

Joe Lanzone, Student, Computer Science, Yale University