AN INTELLIGENT SYSTEM TO IMPROVE VOCABULARY AND READING COMPREHENSION USING EYE TRACKING AND ARTIFICIAL INTELLIGENCE

Harrisson Li[1], Evan Gunnell[2], Yu Sun[3]

[1]Friends Select School, Philadelphia, PA, 19103
[2]3218 Napoli Way, Philadelphia, PA, 19145
[3]California State Polytechnic University, Pomona, CA 91768

ABSTRACT

When reading, many people frequently come across words they struggle with, and so they approach an online dictionary to help them define the word and better comprehend it. However, this conventional method of defining unknown vocabulary seems to be inefficient and ineffective, particularly for individuals who easily get distracted. Therefore, we asked ourselves: "how we could develop an application such that it will simultaneously aim to help define difficult words and improve users' vocabulary while also minimizing distraction?". In response to that question, this paper will go in depth about an application we created, utilizing an eye-tracking device, to assist users in defining words, and enhance their vocabulary skills. Moreover, it includes supplemental materials such as an image feature, "search" button, and generation report to better support users' vocabulary.

KEYWORDS

Eye-tracking, Artificial Intelligence, Vocabulary, Reading Comprehension.

1. INTRODUCTION

The background of my topic arose from the fact that my reading comprehension skills are so poor due to the struggle in vocabulary. When reading an article or book, I constantly must go back and forth between the book and the online dictionary to help me better understand the words in the context. I knew this issue didn't only pertain to me; therefore, I created an app to help individuals, particularly students, improve their vocabulary skills whether it be for educational purposes or just in their daily life. Currently, similar dictionary tools are popularizing and frequently used amongst high school students because humanities teachers often assign students challenging comprehension texts to read.

Some existing tools for this topic include the most basic: going on to the internet to look up the definition of the word. Another tool is this chrome extension created by GoodWordGuide.com called "Instant Dictionary". This extension allows users to double click on a word using their cursors—on any website—they are struggling with, and a mini dictionary will pop up on the same exact screen. There is an app called "Quick Dictionary" which is like Instant Dictionary by the fact that the definition also appears on the same screen. However, the app has an additional

David C. Wyld et al. (Eds): COMIT, CRBL, BIOM, WiMNeT, SIP, AISO - 2021
pp. 27-36, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111703

feature of allowing users to study words they previously searched which is designed to reinforce the user's understanding of the word.

Using online dictionaries is currently the most popular or classic method people use to approach unknown words they come across. The method is generally time-consuming if one wants to find a good definition for their word by checking several websites. Additionally, it's very distracting to readers' focus when they are reading an online article or book. The "Instant Dictionary" extension just doesn't have a very good dictionary, especially for very complicated words, which is really bugging because you would have to look the word through online dictionaries. The "Quick Dictionary" app is only found in the google play store, meaning it is limited to android users, particularly those using the phone. Similarly, the "Quick Dictionary" is also equipped with a less reliable dictionary.

Our application incorporates an eye-tracking device, or simply using the cursor, that provides a side-by-side dictionary with some sort of book or article the user is reading. For that to even happen, users must use either the eye-tracking device or cursor to pause on the word they are struggling with for a specific amount of time to signal that they are having trouble on that particular word. Features of this app help provide definitions from a good-quality, popular dictionary, alongside it carries images (only if the word has one) for an even better understanding of the word. Furthermore, if the dictionary already provided doesn't actually give users a "not so good" definition of the word, there is a "search" button that allows for users to look up the word by providing other dictionary website URLs. This product compared to the other mentioned is simple and efficient. First, users don't need to look up any definitions on the internet because a satisfying definition is most likely already provided; however, if that isn't the case, then it's just simply finding other URLs through the search button. Therefore, the app not only brings forth great definitions but does it in a very efficient way. Users won't have to be concerned about easily being distracted since the dictionary is side by side with whatever the user is reading. All in all, this app is a major progression, especially with the inclusion of an eye-tracking device, making user-experiences much finer compared to that of similar tools/methods!

Successfulness => Yes, the app is not only able to track and define what words users are struggling with, but also provide supplementary materials that can enhance their understanding of those words. Some of the supplementary materials that are implemented consist of an image feature that is complementary to the definition, a "search" button, and a generation report. The additional image feature is provided to reinforce users' understanding of the word's definition. (A quick disclaimer: the image displayed may not always be an accurate representation of the word's definition.) Similar usage to the image feature is the "search" button. This magical button designates users to a dictionary webpage, Merriam Webster, if they are sincerely struggling to understand the word's definition after having read the initial definition and seen the image. Lastly, the generation report is a report on the words users struggled on— meaning a definition and image appeared when a word was either stared at/hovered on for two seconds— throughout the duration users use the application. The report appears after the application is closed by users, and aims to help them efficiently study for the words they struggled with so the words don't become a barrier the next time they come across it.

The application is very successful as it meets the goal, we initially set out to accomplish: using an eye-tracking device to assist in defining challenging words for users. Furthermore, it incorporates additional features that look to increase user experience while improving and enhancing users' vocabulary in reading.

2. FORMAT GUIDE

The rest of the research paper is organized as follows: Section 3 will go in depth about the challenges that we faced while creating the program/app on different coding platforms; Section 4 focuses on the details of how we overcame and resolved the corresponding challenges mentioned in Section 3; Section 5 presents the specifics about the experiment we completed, following with the related work in Section 6. Finally, Section 7 gives the conclusion remarks, and states how I will continue to improve upon the app I created in the future.

3. CHALLENGES

Challenge 1. Figuring out which words are problem words. We don't know when users are struggling with a word because many people aren't outspoken when in such a situation. Furthermore, the goal of our program is to help users define words they struggle with, so it is vital that our program knows that information. When users usually encounter challenging words, they are given access to either an online or physical dictionary alongside the reading. To make the solution more efficient, we designed the application so that when users hover over a word they are struggling with for exactly two seconds, it will process that information and define the word.

Challenge 2. Text and Image Recognition (Natural Language Processing) often takes time to run, hindering the goal of the program. Very frequently, we see many programs that take an extensive time to load so that they may properly function when users utilize the application. This extended wait time shifts the user's attention away where they will find a similar functioning application with a shorter load time. In our application we used Amazon Web Services' Textract which, though relatively fast, still requires time to preload the textract. This time leads to a situation that is not an optimal user experience. A very widespread solution when dealing with this kind of problem is designing a splash screen that would capture the user's attention, psychologically making them feel like they aren't having to wait for the program to run. Similarly, our solution was to implement multithreading as well as a splash screen to overlap and conveniently pace the time needed to run the textract in the background.

Challenge 3. Creating the UI and application to be as quick and smooth as possible, so as to encourage the user to utilize it. Something that is supposed to assist users in their reading, needs to be quick and responsive so that it doesn't become more of a hindrance. You can move on to it specifically being a challenge. We want this program to help user's learn words faster and more efficiently. If the overall program/ UI is too slow, the program ends up costing them more time. Making the program aesthetically simple/clean as well as very responsive encourages users to use it. We designed the program to have a very simple design with very obvious ways of utilizing it. Simply looking at something or tracking your mouse over it, does the lookup for you. We also used the Python Tkinter library since it is lightweight and usable on most computers.

4. METHODOLOGY/SOLUTION

A. Overview of the Solution

An overview of the system is depicted in the figure below. Users simply run the e- Dictionary application on their desktop while reading. Depending on their comfortability and accessibility, they can choose to either hover over words with their mouse cursor (for those who do not have the eye-tracking software) or an eye-tracking device. Regardless, it will display the word's definition and image to the users. Finally, after users finish doing their reading, a generation

report, which gathers the words the user struggled with for the duration of using the application, will appear to further assist them with their vocabulary learning.

B. Step/Components in this System

In order to provide a more efficient setting for users' readings and the potential definitions they seek, we designed an application in Python Tkinter that utilizes the Tobii eye tracking hardware. The application provides a Graphical User Interface (GUI), which was used to give the application a general landscape from viewing the definitions to navigating pages akin to an eReader. After completing the GUI, we implemented Amazon Textract to request the word image recognition and labelling from their site to save to our application. As previously mentioned, there are two modes for word identification: one that tracks the user's mouse location to simulate eye tracking and one that receives an eye tracking position. Depending on our decision heuristic (if they hover too long) we use a dictionary library, web page request library, and imaging library, to pull the appropriate definition, image, and relevant searches. Our multithreaded application will poll (check) for any changes to the user's eye position and update it accordingly in the app. Furthermore, we also preload the app during the splash screen. In order to include the Tobii eye tracking hardware for positioning, our eye Tobii Tracking software (in C# and Unity) will constantly save the user's current eye tracking position. When everything successfully operates, all information is displayed back to the user.

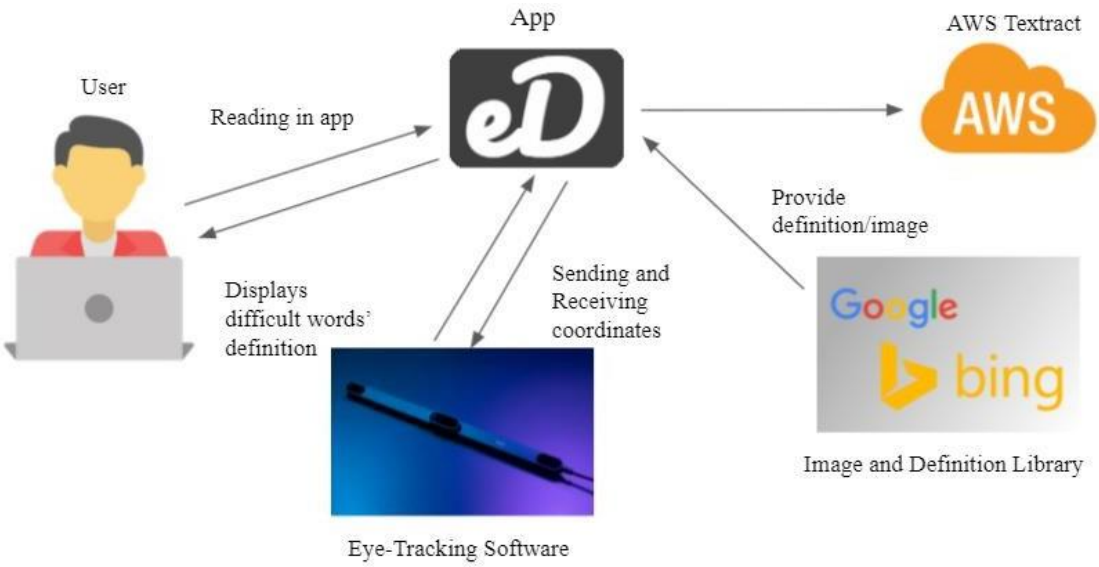

Figure 1. Overview of the Solution

C. Each Component

Python Tkinter is a GUI development library built into Python that allows for simple grid-based app development. The GUI development allows users to navigate pages akin to an eReader using the forward/backward buttons. Users can also view the definitions and images for a specific vocabulary through the inclusion of a dictionary and image library. Furthermore, it increases user experience by giving users the option to click a "search" button, bringing them to a designated dictionary webpage, Merriam Webster, if they have further confusion with a word. After designing the application's overall structure in Python Tkinter, we implemented AWS Textract. AWS Textract or Amazon Textract is an Amazon Web Service that takes in pdfs or other pages and extracts the text into bounding boxes for each word. It utilizes Optical Character Recognition

(OCR) to scan through the documents and provide a quick result of the words to the user. Therefore, using either the mouse or eye-tracker location to identify which words users are struggling on, we check their x, y coordinates (starting from the top left) to see which image from the textract library they are focused on. If users use the eye-tracking software, our Tobii Eye-Tracking software (in C# and Unity) will constantly save the user's current eye tracking position. The Tobii Eye tracker uses the Tobii API from the Tobii Unity SDK. It provides highly accurate monitoring of the user's eye positioning. The eye tracker itself grabs the position by monitoring eye and face position along with IR light to help it more accurately see. The API provides the "gaze data" which is the x, y coordinate of the user's gaze relative to the screen. We then save it to a file at a constant rate. To improve the practicability of our application, we inserted multithreading to the code, which will poll (check) for any changes to the user's eye position and update it accordingly in the app. Additionally, it consists of a splash screen that creates an animation while the application is running in the background to increase the overall speed, resulting in better user experience. For the thread reading the eye tracking software, we are using a library called Watchdog that is able to constantly check for file modifications and updates in a specific sub folder.

```
curr_def_word = ""


def check_when_chosen():
    global curr_def_word
    global word
    global prev_word
    if not eye_tracking:
        time_selected = 0
        while True:

            print("Getting the current word ", word)

            time.sleep(0.5)
            if word == prev_word and word != "":
                print("Hovered over {} for {} seconds".format(word, time_selected))
                time_selected += 0.5
                prev_word = word
            else:
                time_selected = 0
                prev_word = word

            if time_selected >= 1:
                get_word_definition()
                curr_def_word = word
                time_selected = 0
    else:
        time_selected = 0
        while True:
            print("Getting the current word ", word)
            time.sleep(0.25)
            if word == prev_word and word != "":
                print("Hovered over {} for {} seconds".format(word, time_selected))
                time_selected += 0.5
                prev_word = word
            else:
                prev_word = word
```

Figure 2. e-Dictionary coding

```
def get_eye_coordinates():
    global eye_tracking

    global eyeUpperLeft
    global eyeUpperRight
    global eyeBottomLeft
    global eyeBottomRight

    global curr_image_rootx
    global curr_image_rooty
    global curr_image_height
    global curr_image_width

    global window_width
    global window_height

    bbox_UL = [curr_image_rootx, curr_image_rooty]
    bbox_UR = [curr_image_rootx + curr_image_width, curr_image_rooty]
    bbox_BR = [curr_image_rootx, curr_image_rooty + curr_image_height]
    bbox_BL = [curr_image_rootx + curr_image_width, curr_image_rooty + curr_image_height]

    eye_tracker_width = int(eyeUpperRight[0]) - int(eyeUpperLeft[0])
    eye_tracker_height = int(eyeUpperLeft[1]) - int(eyeBottomRight[1])

    while True:
        avgx_location = []
        avgy_location = []
        for i in range(4):
            time.sleep(0.25)
            print("GRABBING EYE DATA")
            #Fetching True Eye Position Data
            f = open("C:/Users/harri/OneDrive/Desktop/New folder/data.txt")
            data = f.read().replace("(", "").replace(")", "")
            data = data.split()[0:2]
```

Figure 3. e-Dictionary coding

5. EXPERIMENT

Author names are to be written in 13 pt. Times New Roman format, centred and followed by a 12pt. paragraph spacing. If necessary, use superscripts to link individual authors with institutions as shown above. Author affiliations are to be written in 12 pt. Times New Roman, centred, with email addresses, in 10 pt. Courier New, on the line following. The last email address will have an 18 pt. (paragraph) spacing following.

A. Does e-Dictionary solve the problem of assisting users?

Our application showcases extremely quick response times for bringing up the appropriate data that users are expecting. The responsiveness of the application dramatically reduces the plentiful distraction towards users, particularly younger kids, when reading. Furthermore, it can determine what the user is struggling with through the decision heuristic: if they sit on a word for 2 seconds. In our own testing, this application had a positive impact on vocabulary-comprehensive reading. Additionally, it carries out useful data from the generation report to reinforce vocabulary learning for users, hoping it will make reading easier for users as they are exposed to more vocabulary (and their definitions) in the long-term.

B. Are we able to integrate eye tracking to an application like this? Does it work?

Yes, we were able to integrate eye-tracking into our application through using the Tobii API from the Tobii Unity SDK and Watchdog library. Unfortunately, due to the inconsistency of the Tobii eye-tracking software, the device does not work well with the eDictionary application. When

experimenting with the eye-tracking software several times, we noticed that the coordinates displayed from the eye-tracking software of the word were incompatible with the coordinates displayed when utilizing the mouse cursor. This posed a challenging problem for us, so we decided to have users, for now, use the traditional mouse cursor as they wouldn't frequently encounter problems with the eye-tracking software.

C. Did the final program meet the requirements we set out at the beginning of the paper? Did we manage to accomplish anything?

The implementation of an automatic eye-tracking device to the application did indeed meet the needs of the problem we were looking to solve. It is able to analyze specific words the user wants to comprehend; using that information, the remainder of our application will utilize Amazon Textract to provide users with both a visual and text-based definition to the challenging vocabulary, instantly. Overall, our application is able to successfully provide not only assistance in a difficult reading language, but doing so while being fully automated. Through our own testing, this application serves as a wonderful digital assistance to those looking to learn complicated words in new fields or even in a new language, English. The ease of not needing to do anything beyond reading the text while still maintaining active searching of complex words truly helps users with their readings.

D. Summary

The responsiveness and high user experience of e-Dictionary effectively undermines user's quirk of getting frequently distracted while looking up words on a dictionary application. Furthermore, with the inclusion of an eye-tracking software, it further reinforces the efficiency of users having to look up very demanding words and also improves their vocabulary. All in all, the attributes of this application as mentioned above, do satisfy the criteria of the problem we initially set out to solve.

E. Data

	Period 1				
Eye-tracker	16		Mean	Median	Standard Deviation
Mouse Cursor	19		14.66666667	16	5.131601439
Manual	9				

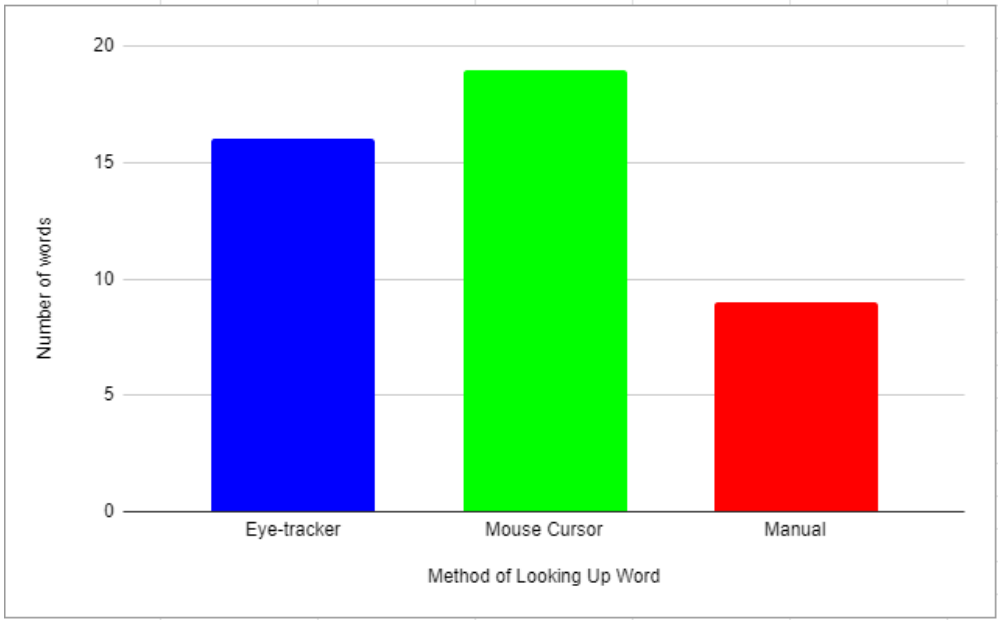

Figure 4. Number of words looked up in 5 minutes.

	Period 1				
Eye-tracker	5		Mean	Median	Standard Deviation
Mouse Cursor	4		6	5	2.645751311
Manual	9				

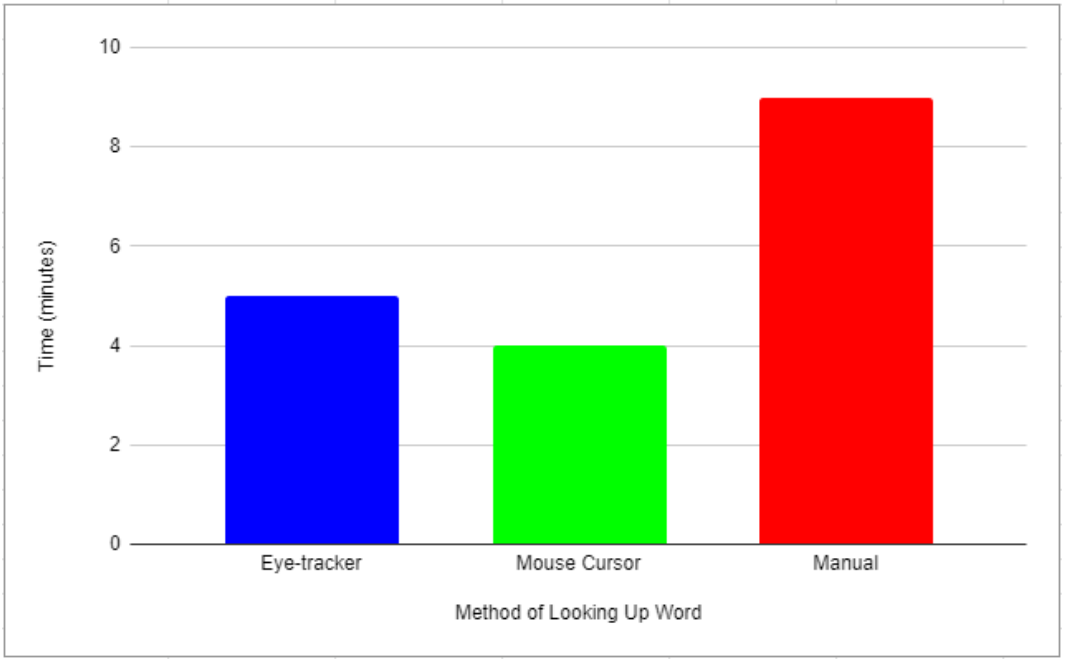

Figure 5. Time taken to get through one SAT Reading passage with challenging vocabulary.

6. RELATED WORKS

Related Work 1. Katsuyu Fujii and Jun Rekimoto created an application called SubMe, a smart subtitle system with a machine learning algorithm for estimating users' English levels. The goal of this application was to estimate English levels through using an eye tracker and gathering eye gaze movement data. Unlike my work which was designed to help improve user's English skill, specifically in concentration of their vocabulary, Subme was used to estimate English levels through using a smart subtitle system and eye tracking data. One strength of this application is being able to gather data, which hypothetically is more convenient when trying to improve a problem than that of an application that primarily aims to help improve a user's English.

Related Work 2. Elizabeth Krupinski and Josh Borah researched how the use of an eye tracking technology correlates with the improvements of speed and reading of radiological readings. They concluded that there was a positive correlation between the two; different eye tracking systems had the ability to track, record, and analyze eye movement which not only improved image reading but also could easily detect flaws during the radiological readings. My work was geared towards creating an application, while Elizabeth and Josh researched how eye tracking technology helped improve radiological readings. However, both our work demonstrated the benefits of an eye tracking technology's functionality in a variety of fields.

Related Work 3. A group of 6 people developed an application called Private Reader; it implements an eye tracker that shows only the portion of text that the user is focused on while reading. The goal was to maintain privacy from nosy people in public work spaces while the users were reading. The application created by the group, I would say, is very close in difficulty to the application I created. Furthermore, their application in some ways is definitely better than mine. For example, one strength their work acquired was having 6 people collaborate together on the project, which did not allow for more ideas but also higher efficiency when creating the application. Additionally, before creating the application they did a user study that would give users a better user experience, which is beneficial when promoting and selling the products to others.

7. CONCLUSION AND FUTURE WORK

In this project, we had the idea of assisting people who struggled with vocabulary in their daily lives. One of the main problems with learning vocabulary is that of actually looking up and understanding the definition of a word. The deeper one delves into more complex readings at a greater level of education, the more frequent one will pause to look up the definition of a word. Therefore, to ameliorate this problem, we designed an application that not only will make looking up the word's definition faster for users, but also automate the process for them. We do this in two different ways: 1. having the definition pop up for the user on the side when they hover over a word with the eye-tracking device 2. having the user click the "search" button, bringing them to a dictionary webpage, Merriam Webster, if they have further confusion for the word. Through frequent experimenting with the application, results show that it is very successful. Although the project is not perfect, it carries many features that make it unique from other dictionary webpages or applications, such as implementing an eye-tracking device, providing a combination of images and definitions, and equipping a "search" button that offers an alternative for looking up definitions.

One big limitation of the project is the accuracy of the image API, which is supposed to increase efficiency of vocabulary learning for users. However, it often provides images that are not compatible with the word users are struggling with, resulting in further confusion for them.

Another limitation is the practicability of the eye-tracking application; whether it's the splash screen or tracking of a word, a good portion of the application remains to be somewhat choppy when users are using it. Finally, even though the optimization of the application is definitely not the greatest (as there could be more features added to it to make it seem more multifaceted), it is certainly usable for users.

In the future, I hope to find a better image API to implement into the application that will result in better user experience. Furthermore, I hope to increase the responsiveness and speed of the application by executing future touches to the programming. Last but not least, expanding the use cases to improve user experience, such as adding features like translations of words, analyzing data for users' vocabulary, a thesaurus accompanied by the dictionary, all will require further programming and time dedicated to the application.

REFERENCES

[1] Engdahl, Sylvia. "AWS Machine Learning Blog." Amazon, Greenhaven Press/Gale, 30 May 2019, aws.amazon.com/blogs/machine-learning/automatically-extract-text-and-structured-data- from-documents-with-amazon-textract/.

[2] Developers, Tobii. "Unity Sdk." Tobii Developer Zone, 14 July 2021, developer.tobii.com/pc-gaming/unity-sdk/.

[3] Amos, David. "Python Gui Programming with Tkinter." Real Python, Real Python, 3 Apr. 2021, realpython.com/python-gui-tkinter/.

[4] Khandelwal, Renu. "Monitoring Your File System Using Watchdog." Medium, Analytics Vidhya, 11 Jan. 2021, medium.com/analytics-vidhya/monitoring-your-file-system-using- watchdog-64f7ad3279f.

[5] Katsuya Fujii and Jun Rekimoto. 2019. SubMe: An Interactive Subtitle System with English Skill Estimation Using Eye Tracking. In Proceedings of the 10th Augmented Human International Conference 2019 (AH2019). Association for Computing Machinery, New York, NY, USA, Article 23, 1–9. DOI:https://doi.org/10.1145/3311823.3311865

[6] Krupinski, E., & Borah, J. (2006). Eye tracking helps improve accuracy in radiology. Biophotonics International, 13(6), 44-49. https://arizona.pure.elsevier.com/en/publications/eye- tracking-helps-improve-accuracy-in-radiology [3] Amos, David. "Python Gui Programming with Tkinter." Real Python, Real Python, 3 Apr. 2021, realpython.com/python-gui-tkinter/.

[7] Kirill Ragozin, Yun Suen Pai, Olivier Augereau, Koichi Kise, Jochen Kerdels, and Kai Kunze. 2019. Private Reader: Using Eye Tracking to Improve Reading Privacy in Public Spaces. In Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '19). Association for Computing Machinery, New York, NY, USA, Article 18, 1–6. DOI:https://doi.org/10.1145/3338286.3340129

[8] Mehvish, Mehvish. "7 Best Dictionary Extensions for Chrome." Guiding Tech, 22 June 2020, www.guidingtech.com/best-dictionary-extensions-chrome/.

[9] "Quick Dictionary – Apps on Google Play." Google, Google, 18 July 2020, play.google.com/store/apps/details?id=com.yaki.wordsplash&hl=en_US&gl=US.

AUTHORS

Hi! I'm **Harrisson Li**, the inventor and creator of e-Dictionary. I'm a high school senior that resides in Philadelphia; I'm passionate about basketball and math. When being asked about problems that I face in my daily life, I immediately thought of my poor vocabulary skills and lazy personality. From there, I thought of creating something that would make using dictionaries more efficient for people. With my enthusiasm in technology and programming, I decided to create e-Dictionary.