A POWERFUL CHROME EXTENSION: TRANSLATION PROGRAM USING PYTHON, WEBSITE ANALYSIS AND GOOGLE FIREBASE SERVICES

Tongde Zhao[1], Khoa Tran[2]

[1]Santa Margarita Catholic High School, 22062 Antonio Pkwy,
Rancho Santa Margarita, CA 92688
[2]Computer Science Department, California State Polytechnic University,
Pomona, CA 91768

ABSTRACT

The last few decades have seen a remarkable development technological wise in order to provide understanding between people with different language backgrounds. This paper develops a Chrome Extension that scans texts on a website and discovers unfamiliar English words to its users, then translates those words and displays their meanings onto their computer screens. Using Python with Flask framework and Google Firebase as a backend means, this Chrome Extension can be downloaded on any Chrome Browser and provide definitions to difficult words on websites that users want to get translated. The extension is useful to those who are having a hard time understanding difficult English vocabulary. In the experiment, the application has shown to detect 95 to 98% of the difficult words users are struggling with, with the speed of 10 minutes for documents that have more than 2000 words. The results show that the application provides adequate predictions for most users. For this application, I took data reliability, usability, and web-scrapping into consideration. The extension required ample reliable data, user-friendly interface, and a stable Internet browser for usage.

KEYWORDS

Language Translation, Firebase Database, Python, Web Analysis

1. INTRODUCTION

Learning English is an important endeavor in this ever changing world [1]. English is widely spoken for 250+ million people as a mother tongue, with a further 100 million using it as a second or foreign language[12]. Furthermore, English is considered to be the international language of business, science, technology and education [3][4][5][6]. Mastering English provides the opportunity to communicate and collaborate with people from all over the world as many companies make English proficiency as a job requirement or at least an ideal candidate's quality, especially in fields such as international business, marketing and technology [7]. However, for many English learners, especially to those who speak English as their second language, they struggle with demanding study tasks such as keeping up with required reading, completing longer writing tasks, and undertaking critical and conceptual analysis [2]. In America, those with Spanish background found it difficult to understand mathematical vocabulary at the college level [8]. Another problem for English-as-a-second-language learners is that they lack the keyword vocabulary, thus putting them in further disadvantage when it comes to absorbing new

David C. Wyld et al. (Eds): ARIN, CSITA, ISPR, NBIOT, DMAP, MLCL, CONEDU, NATAP -2023
pp. 179-189, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130716

knowledge in a more English professional setting [9]. Therefore, the demand for reading and understanding good and proper materials on legitimate news resources is becoming more and more important to those learners' learning process as reading newspapers absorbs information in a different way than reading scientific journals [10]. Thus for those who are learning English, a dictionary hand-in-hand while reading is an invaluable resource so that they can look up those difficult keywords quicker and, consequently, understand the source materials faster [11].

I have conducted an experiment in which they developed an extension to run and translate difficult words extracted from five different websites with varying levels of writing styles and difficulty. The experiment aimed to determine the effectiveness of the program in extracting difficult words from documents, and to identify any potential blind spots. I have used an Excel chart to compare the amount of difficult words extracted by the program and the actual percentage of hard words extracted by traditional means. The results showed that the program extracted a similar amount of difficult words as users using more traditional methods. I then selected five scientific articles with more than 2000 words for the extension to test run. The experiment recorded the time it took to scrape and translate the difficult words, which were then displayed in two charts. The results showed that there were areas to improve the program's execution time, and that the more difficult the document, the longer it took for the program to run. Overall, the experiment demonstrated the effectiveness of the program in extracting difficult words, but also identified areas for improvement.

I propose an application that will allow users to easily translate difficult words directly from websites through Chrome, Firefox,.. extensions. Using open data from Google Scholars and several news articles from respected sites like New York Times or Bloomberg – the extension will detect difficult vocabulary from a specific article, translate these words, and display them on the user's screen through web interface. Users can choose any articles they want for the extension to scan, the program will analyze and produce a list of potential difficult words that users may have a hard time understanding. Showing the specific definitions and their categorizations would make it easier for English learners to read and understand source materials better and in a faster time frame. The extension that I developed contains a list of unfamiliar words compiled from multiple academic English articles like Google Scholar, New York Times or Nature. The list will be hosted inside a cloud database provided by Google Firebase and will be accessed by API calling mechanisms from the Python codebase. This is an effective solution because the use of a cloud hosting service allows me to quickly and sufficiently analyze and detect unfamiliar words scraped from the online article and translate these words to the users. Additionally present within my proposed extension is a web interface hosted by Flask server that shows the words being translated. This allows users to see the vocabulary one may find difficult so that when they start reading the article, they will have an easier time understanding the material.

There have been various web scraping methods used to extract texts from websites, categorizing and then translate those words. Kurtis Clarke used an alternative to WebScraper3000, a popular Python web scraping library, to collect data from websites of his choice, but the program had some drawbacks, including limited information gathering and the need for tutorials to work on certain websites[13]. Clarke then proposes using the open-source BeautifulSoup4 library for web scraping, which can extract a larger amount of information in a shorter amount of time. Tamas E. Doszkocs also developed an associative interactive dictionary (AID) system for search strategy formulation, but the system is unreliable and time-consuming when tested under long documents[14]. The Oxford Dictionary online edition is recommended for English learners, but it requires users to input specific words[15]. Overall, my Chrome Extension is an improvement upon existing English-assistance programs that are available now. I believe that the advantages of using a Chrome Extension to automatically scan and extract difficult words with definitions and

word types, is unmatch, and that it has shown to be great convenient for those who are struggling with understanding English in text form.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The ability to take down raw words

One of the main components of my reading extension is the ability to take down raw words. This feature allows users to finish the text they are reading more hilariously, making it easier for them to understand and retain the information. There are several issues to consider when implementing this component. First, text-to-word requires a lot of processing power, which can slow down the performance of the extension, especially on older computers. Second, a text may have problems with the same word appearing multiple times, which may negatively impact the user experience. To solve the problem of the large amount of processing power required for text-to-phrase, I can use cloud-based technology to move the processing power to the cloud, which results in smoother performance on the user's device. For the problem that the same word may appear multiple times in a text, caching techniques can be used to store the words that have already been processed to avoid duplicate processing. In this way, processing time can be reduced and smooth performance can be guaranteed.

2.2. Expanding Vocabulary

Expanding vocabulary is a major component of my reading extension. When implementing this feature, several problems must be considered. Firstly, ensuring that the words suggested for expansion are relevant to the user's reading level and interests. Secondly, ensuring that the definitions and examples provided for the new words are accurate and easy to understand. Thirdly, the feature must be user-friendly and intuitive to use. To address these problems, I could use a combination of natural language processing and machine learning algorithms to analyze the user's reading habits and suggest relevant words for expansion. I could also use reliable sources, such as dictionaries and encyclopedias, to ensure that the definitions and examples provided are accurate. Additionally, I could design a user-friendly interface with clear explanations and examples, making it easy for users to understand and retain the new words.

2.3. The personalized recommendation system

One major component of my reading extension is the personalized recommendation system. This feature uses user data to suggest articles and books based on their reading habits and interests. When implementing this component, several problems must be considered. Firstly, ensuring that the recommendations provided are accurate and relevant to the user. Secondly, handling the privacy and security of user data. Thirdly, ensuring that the recommendation system is able to adapt and improve over time. To resolve these problems, I could use machine learning algorithms to analyze the user's reading habits and make recommendations based on that data. I could also implement strong privacy and security measures to ensure that user data is protected. Additionally, I could use feedback mechanisms, such as user ratings and reviews, to continuously improve the recommendation system and provide users with even more relevant and personalized recommendations.

3. SOLUTION

The extension uses the Flask web host, which I used as it is robust, expandable, and simple to develop for, especially Python. When the extension is activated when the user picks a certain article from the web, the extension will activate its Python script to scan the article and get all the words in that specific page. The extension has a list of difficult words pre-generated by AI scanning through news articles and scientific journals. The application will then “clean” the article using Python by isolating the words and removing numbers, hyphens, and anything unrelated to English. Next, another Python will read the words provided from the previous step and compare them to the pre-generated list of difficult words. If any of the words in the article match the difficult words list, the extension will highlight those words and provide a definition when the user hovers over them. Additionally, the extension will keep track of the user's progress by logging the words they have looked up and the articles they have read. The logged data will be used to personalize the user's experience by suggesting articles based on their interests and highlighting words that the user has previously struggled with. The ultimate goal of the extension is to help users expand their vocabulary and improve their reading comprehension by making it easier to look up and understand difficult words while reading online. The application will send a signal to a python flask server hosting and display the vocabulary and their definitions or a generic message describing how the website the user picks does not allow the extension to run a scan on.

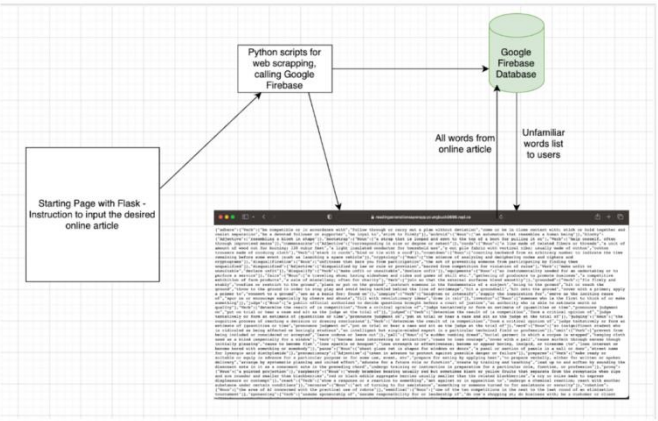

Figure 1. Overview of the solution

The component is a Python script that scans articles and compares the words to a pre-generated list of difficult words. It relies on Flask web host and uses Python to isolate and clean the words. It uses NLP (Natural Language Processing) to identify and highlight difficult words and logs user progress to personalize the experience. The component improves reading comprehension and expands vocabulary by making it easier to understand difficult words while reading online.

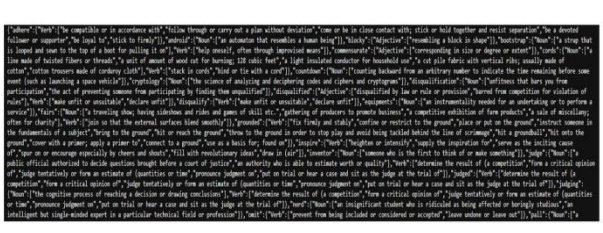

Figure 2. The script

```
def word_definition(word):
    dictionary = PyDictionary()
    definition = dictionary.meaning(word, disable_errors=True)
    if(definition == None):
      return None
    else:
      return definition
    return "Success"

# define all unknown words in a website
@app.route('/scrape_website')
```

Figure 3. Screenshot of code 1

```
def scrape_words(url):
    html = urllib.urlopen(url).read()
    soup = BeautifulSoup(html, 'html.parser')
    soup = soup.body
    text = soup.find_all(text=True)
    output = ''
    blacklist = ['[document]', 'noscript']

    for t in text:
        if t.parent.name not in blacklist:
            output += t+' '
    output = output.split()
    cleanOutput = [word for word in map(clean, output) if word in word_set]
    #print("#################")
    #print("Start to print CleanOutput")
    #print(cleanOutput)
    return cleanOutput
```

Figure 4. Screenshot of code 2

This code section is specifically designed for web scraping, which involves extracting text from a website. To achieve this, a web scraper tool is utilized to enable the code to access and extract data from the website. Once the text is obtained, the code employs advanced algorithms to differentiate between nonsensical words and legitimate English words using an online dictionary directory. The algorithms enable the code to effectively classify the text into two categories based on its linguistic value. This process is aimed at ensuring that only relevant and meaningful data is extracted from the website while filtering out any irrelevant or nonsensical information. By streamlining the data analysis process, this technique can make it more efficient and save time and effort for the user. This can be especially helpful when dealing with large amounts of data. The ability to extract relevant information from a website can be used in a variety of fields such as market research, data science, and competitive analysis. Additionally, the code can be modified to extract specific types of information, such as product prices or customer reviews, to gain insights that can inform business decisions. Overall, web scraping is a powerful tool for data extraction and analysis, and can help businesses and individuals make informed decisions based on relevant and accurate data.

One of the components of the system is a browser extension that extends vocabulary by highlighting difficult words and providing definitions. The system is implemented using Flask web host and Python scripts that scan articles for difficult words. The component relies on a pre-generated difficult words list generated by AI scanning, and does not rely on any special concept. The extension functions to help users expand their vocabulary and improve reading comprehension while browsing the internet.

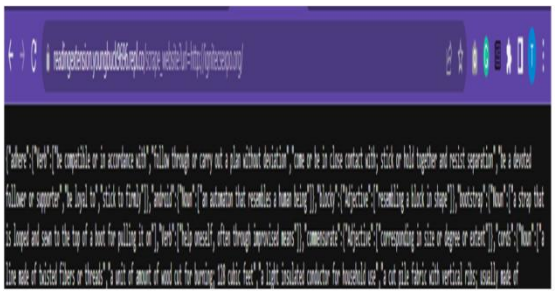

Figure 5. The screenshot of the website

```
@app.route('/scrape_website')
def define_all():
  url = request.args['url']
  definitions = {}
  print(url)
  a = scrape_words(url)
  un_common_word = get_uncommon_word('tskhoa1999@gmail.com', a)
  count = 0
  print(len(un_common_word))
  #print(un_common_word)
  for word in un_common_word:
    print("##########################")
    print(word)
    print("This is " + str(count) + " loops")
    count += 1
    definition = word_definition(word)
    if(definition != None):
      definitions[word] = definition
  return definitions
```

Figure 6. Screenshot of code 3

```
@app.route("/remove_word/<user>/<word>")
def remove_user_word(user, word):
    user_ref = db.collection(u'Users').document(user)
    user_ref.update({u'common_user_words':
firestore.ArrayRemove([word])})
    return word + " is succefully removed"

#add user's common word to their list
@app.route("/add_word/<user>/<word>")
def add_user_word(user, word):
    user_ref = db.collection(u'Users').document(user)
    user_ref.update({u'common_user_words':
firestore.ArrayUnion([word])})
    return word + " is succefully added"

#remove user's unknown word to their unknown list
@app.route("/remove_unknown/<user>/<word>")
def remove_user_unknown(user, word):
    user_ref = db.collection(u'Users').document(user)
    user_ref.update({u'uncommon_user_words':
firestore.ArrayRemove([word])})
    return word + " is succefully removed"
```

Figure 7. Screenshot of code 4

The code defines four Flask routes that implement various functionalities. The define_all() function runs when the user clicks on the browser extension, and it scrapes the text from the given URL using the scrape_words() function. It then identifies uncommon words in the text and retrieves their definitions using the word_definition() function. The definitions are returned as a dictionary to the browser extension. The add_user_word(), remove_user_word(), add_user_unknown(), and remove_user_unknown() functions allow the user to add or remove words to their common or uncommon word lists in the backend Firestore database.

The AI scanning code runs before the program starts and generates a predefined list of difficult words from news articles and scientific journals. The define_all() method represents the functionality of the browser extension, while the other methods communicate with the Firestore database. The url variable holds the URL of the website to be scrapped, and the definitions

variable is a dictionary that holds the uncommon words and their definitions. The server communicates with the Firestore backend to add or remove words to the user's common or uncommon word lists.

One of the components in 3.1 is a word extension that adds new words to the user's vocabulary. This component is implemented using a dictionary API service that provides definitions of new words. It does not rely on any special concept such as NLP or neural networks. The component functions by allowing users to add or remove words from their vocabulary lists in the database.

```
firebaseConfig = {
  "apiKey": "AIzaSyCwQb5jrBTwKeaWBZUcVPQYyW6TEH9IkXk",
  "authDomain": "reading-extension-175ab.firebaseapp.com",
  "databaseURL": "https://reading-extension-175ab-default-
rtdb.firebaseio.com",
  "projectId": "reading-extension-175ab",
  "storageBucket": "reading-extension-175ab.appspot.com",
  "messagingSenderId": "877728621082",
  "appId": "1:877728621082:web:19999e0ed8c428aef327a3",
  "measurementId": "G-DPQPZ62LFD"
};
```

Figure 8. Screenshot of code 5

The code shown appears to be a JavaScript object that contains the Firebase configuration settings for a web application. It likely runs at the initialization of the application to connect to Firebase services such as authentication, database storage, and messaging. Each key in the object represents a different Firebase service, such as authentication with "authDomain", database storage with "databaseURL", and messaging with "messagingSenderId". These keys and their associated values specify the settings necessary for the application to interact with Firebase's backend servers and services. The variables being created in this code are properties of the firebaseConfig object, and they store the settings for each Firebase service that the application will use. For example, the projectId variable contains the unique identifier for the project on Firebase's servers. Overall, this code defines the necessary settings for a web application to connect and interact with Firebase's backend services. The backend server is managed by Firebase and handles tasks such as user authentication and data storage.

4. EXPERIMENT

4.1. Experiment 1

A concern with the Chrome Extension is that the program may not discover enough difficult words to translate . Experimentation with Python web-scraping and translation algorithms can provide an understanding of the running of the code, as well as giving an indicator on areas that need to be improved.

Using 5 different websites with different writing styles and levels of difficulty, I made the extension to run and translate the difficult words extracted from those websites. In order to make sure that the test yields the proper outcome, I have extracted the difficult words from the 5 websites beforehand to keep them as a reference. I then created an Excel chart to illustrate the findings and compare with the amount of difficult words I had prepared before. I believe that the test will give the answer to the possible blind spot since the websites I have chosen represent a diverse range of online resources that users will interact with.

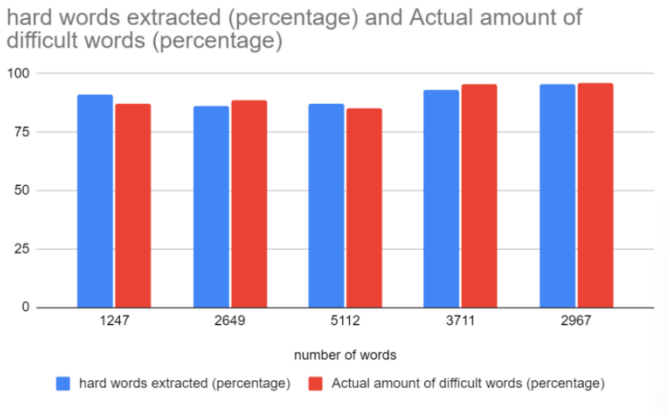

Figure 9. Graph of experiment 1

Figure 9 is a graphical illustration of the percentage of hard words gathered by the program and the actual percentage of hards words extracted by traditional means in percentage. The blue column depicts the percentage of difficult words extracted from documents by the program and the red column depicts the real percentage of difficult words in those documents. The mean different percentage between these two methods is 2.262 percent, while the median difference is 2.4 percent. The highest percentage of hard words detected by the program is at 95.67 % for a 2967 word document, while the lowest percentage is at 86.17%for 2649 word document. The data has shown that there are certain differences between the amount of difficult words extracted from the program or by traditional means, but the percentage difference is not significant and that in this experiment, the program extracted similar amount of difficult words as users using more traditional methods. The effect of this experiment is significant in which the program has demonstrated that it has yielded similar result as users go through the document line by line to detect difficult words.

4.2. Experiment 2

Another concern for the Chrome Extension is its speed. By doing web scraping while actively dividing words into the known and the unfamiliar, the speed in which the extension runs can give insights on how the effectiveness of the app and the area in which the extension can be improved. 5 scientific articles that have more than 2000 words are selected for the extension to test run. The time it requires to scrape all the words in the article and the time it takes to translate the difficult words would all be recorded. Following that, I create an Excel sheet to record the recordings and display them in a form of 2 charts. I believe this experiment can determine whether the web-scraping mechanism and translation run smoothly and effectively. Furthermore, because of the sheer amount of words and levels of difficulty of the selected articles, the extension will be put to the limit and if there are certain errors appear, I can find the blind spot from my algorithm.

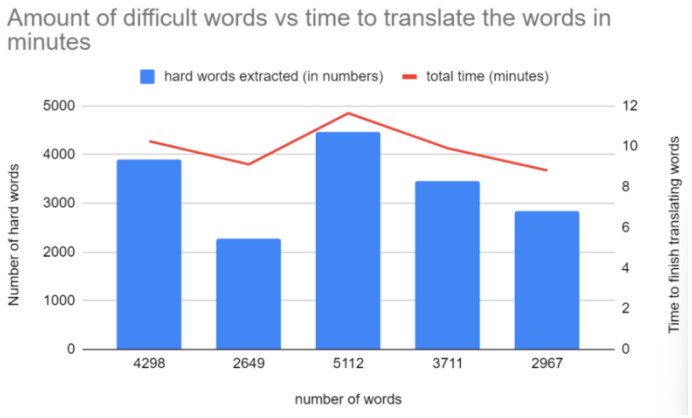

Figure 10. Result of experiment 2

Figure 10 is a graphical illustration of the time, in minutes it takes to scan the document, categorizing the hard words and translate them. The blue column depicts the number of difficult words extracted from documents by the program and the red line depicts the time in minutes it takes to run the program. The mean different percentage between the processing time is 9.95 minutes for documents that contain more than 2000 words, while the median time is 9.91 minutes. The longest time it takes to translate all the hard words is 11.64 minutes for for a 5112 word document, while the shortest time is 9.12 minutes for a 2649 word document. The data has shown that there are certainly areas to improve on the program's execution time, and that the result proves that the more difficult the document is, the longer it takes for the program to run. The effect of this experiment is significant in which the program has demonstrated that there are areas to improve the effectiveness of the program.

5. RELATED WORK

Kurtis Clarke has used an alternative of the popular Python web-scrapping library, WebScraper3000 to web scrape, collect internet data, from sites of his choices[13]. Mr.Clarke manages to scrape through the page and gather information about the documents and the comments from the site's users. However, the program has many setbacks, including that it can work properly only on websites where it has tutorials about scraping them and that the information it scrapes is little. Our method uses a superior library of BeautifulSoup4 by Python to scrape website information, yielding a far more amount of words in a much quicker execution time. Furthermore, since the BeautifulSoup4 is open-sourced, the library is deemed by developers to be safer [16].

Tamas E. Doszkocs proposed Associative Interactive Dictionary. (AID) system for search strategy formulation on a large operational free text on-line bibliographic retrieval system [14]. The procedures rely on statistical frequency distribution information about term occurrences in a set of document texts retrieved in response to a Boolean search query and the occurrence frequencies of the same terms in the entire data base. However, the system is proved to be unreliable when tested under long documents because of its reliance on continuously scanning the entire database, affecting the execution time. In addition, the system is developed when the internet is not yet available, thus hindering its capabilities to be delivered to people worldwide.

Oxford Dictionary, online edition is proposed to be a good resource for any new English learners to have an online dictionary within their grasps[15]. Researches deduce that the online Cambridge dictionary influences student pronunciation and vocabulary in terms of pronunciation

mastery and vocabulary enrichment. Nonetheless, my Chrome Extension has an edge: the Oxford Dictionary can only look up words that the users need to input certainly will frustrate many new English learners. On the otherhand, my program uses auto scanning and algorithm to find difficult words, scanning them and then deliver a list of difficult words with definitions and word types.

6. CONCLUSIONS

The Chrome Extension has some problems that need to be dealt with. While the experiment demonstrated the effectiveness of the program in extracting difficult words from documents, there are several limitations to the program. Firstly, the program relies on the accuracy of the web-scraping mechanism and the translation software, which may not always be reliable. Additionally, the program may struggle with certain writing styles or languages, as it may not be able to accurately identify difficult words. Secondly, the program's execution time can be lengthy, particularly for longer and more difficult documents, which may limit its practical use for users who require quick translations. Lastly, the program only identifies difficult words, and does not provide any context or explanations for these words, which may limit its usefulness for language learners who require a deeper understanding of the language. Overall, while the program is useful for identifying difficult words in documents, it has limitations that may impact its practical use in certain situations.

To solve the limitation of the scraping process requiring regular upkeep, improve scanning algorithms to read multiple types of online resources; I can also improve the detection mechanism for difficult words by infusing AI-based user behavior detection, in such a way that regular upkeep is either not necessary or would not need to be done as frequently.

REFERENCES

[1] Farrall, Cate, and Marianne Lindsley. Professional English in use: marketing. Cambridge University Press, 2009.
[2] Read, John, Catherine Elder, and John Read. "Post-entry language assessments in Australia." Assessing English proficiency for university study (2015): 25-46.
[3] Kang, Hyeon-Seok. "'BTS, PSY, BLACKPINK': The English naming of South Korean music entertainers–its prevalence and motives." English Today: 1-11.
[4] Jia, Mian, and Yi An. "Language as an interpersonal marker in English dissertation acknowledgments: Variations across genres and academic disciplines." English Today: 1-8.
[5] Modiano, Marko. "The future of British English in the European Union: What standard will the EU adopt in the post-Brexit era?." English Today (2022): 1-6.
[6] Tsui, Amy Bik May, ed. English language teaching and teacher education in East Asia: Global challenges and local responses. Cambridge University Press, 2020.
[7] O'Carroll, M., & Kimbro, S. (2021). Legal Operations at Google. In D. Katz, R. Dolin, & M. Bommarito (Eds.), Legal Informatics (pp. 501-510). Cambridge: Cambridge University Press. doi:10.1017/9781316529683.034
[8] Lesser, Lawrence M., and Matthew S. Winsor. "English language learners in introductory statistics: Lessons learned from an exploratory case study of two pre-service teachers." Statistics Education Research Journal 8.2 (2009): 5-32.
[9] Lawrence, Joshua F., et al. "Generating vocabulary knowledge for at-risk middle school readers: Contrasting program effects and growth trajectories." Journal of Education for Students Placed at Risk (JESPAR) 19.2 (2014): 76-97.
[10] Flavián, Carlos, and Raquel Gurrea. "Reading newspapers on the Internet: the influence of web sites' attributes." Internet research (2008).
[11] Kondal, Bonala. "The benefits of using dictionary skills among the third year pharmacy students." International Journal of Management and Social Sciences Research 7.11 (2018): 1-6.

[12] Bolton, Kingsley, and John Bacon-Shone. "The statistics of English across Asia." The handbook of Asian Englishes (2020): 49-80.
[13] Elgendy, Nada, and Ahmed Elragal. "Big data analytics: a literature review paper." Advances in Data Mining. Applications and Theoretical Aspects: 14th Industrial Conference, ICDM 2014, St. Petersburg, Russia, July 16-20, 2014. Proceedings 14. Springer International Publishing, 2014.
[14] Doszkocs, Tamas E. "AID, an associative interactive dictionary for online searching." Online Review 2.2 (1978): 163-173.
[15] Ambarwati, Rosita, and Berlinda Mandasari. "THE INFLUENCE OF ONLINE CAMBRIDGE DICTIONARY TOWARD STUDENTS'PRONUNCIATION AND VOCABULARY MASTERY." Journal of English Language Teaching and Learning 1.2 (2020): 50-55.
[16] Robillard, Martin P., Wesley Coelho, and Gail C. Murphy. "How effective developers investigate source code: An exploratory study." IEEE Transactions on software engineering 30.12 (2004): 889-903.