ASLConnect - A Comprehensive ASL Visual Dictionary: Report

ADITYA POTLURI, Georgia Institute of Technology, USA
MAY KALNIK, Georgia Institute of Technology, USA
SMRITHI NARAYAN, Georgia Institute of Technology, USA
HARI KRISHNA KUMARAN, Georgia Institute of Technology, USA

Additional Key Words and Phrases: ASL, Dictionary, iOs, Android, Google Chrome Extension, Google Docs Add On

ACM Reference Format:
Aditya Potluri, May Kalnik, Smrithi Narayan, and Hari Krishna Kumaran. 2024. ASLConnect - A Comprehensive ASL Visual Dictionary: Report. 0, 0, Article 1 (October 2024), 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Abstract

This report discusses the formulation and creation of a robust, cross-platform American Sign Language (ASL) translation dictionary. Our goal was to not only create an accurate, intuitive, and comprehensive dictionary, but also make it ubiquitous in the people's lives by implementing it across multiple platforms- namely an iOS app, Android app, Chrome extension, and Google Docs add-on. We discuss current and past literature in identifying the complexities, necessity, and feasibility behind constructing a functioning ASL dictionary. Drawing on this literature, we addressed challenges such as context-based translation, optimized data retrieval across platforms, and ease of use. We then discuss our accomplished deliverables and the process it took to implement each platform. Finally, we conclude with a reflection on the results and future work to be completed.

2 Introduction

The ability to hear can often be a barrier in communication between the deaf and hearing communities. This limits lots of potential in social and productive conversation. There needed to be a accessibility tool in which people can do quick translations in order for the communities to talk to each other. In one example, [6] details how the deaf community is widely underrepresented and unfacilitated in the STEM field. The paper describes how a combination of institutional, societal, and language-specific barriers allow the continued loss of potential talent from the deaf community. Relevant to our problem, Mahajan et al. narrates how current video conferencing platforms do not have the means necessary to fully support deaf users; in short, the hearing and deaf users have widely contrasting experiences on these platforms. The underscores the importance of creating Sign Language (SL) centered tools and interfaces not only in functionality but also in terms of design, namely layout and navigation.[6].

It's evident that SL (and by extension, ASL) capabilities in existing technology need improvement. Our project aimed to contribute to this effort, communcation between the deaf and hearing through our robust dictionary implementation. To ease the process of using and learning SL, our objective was to create an American Sign Language (ASL) dictionary that is ubiquitous and consistent across a wide variety of platforms. Namely these platforms are:

- iOS application
- Android Application
- Chrome Extension
- Google Docs Add on

In doing so, we integrated the process of using and learning of sign language seamlessly the lives of anyone who wishes to learn ASL. Our secondary objective was to make this ASL dictionary as robust as possible and provide accurate translations consistently considering context, grammar, and homonyms. Further research revealed components to an ASL dictionary that would be paramount to achieving this goal.

The primary challenge of an ASL dictionary is the translation of text to ASL. While our research demonstrated a wealth of options for the translation of ASL to text that occurred due to advances in computer vision, the translation of text to ASL remains at the frontier of machine learning research [2]. While options exist to convert it to ASL's glossing (which basically breaks down words into their individual morphemes), a natural text to ASL translation would require rendering images from text input using context (a problem which may eventually be tackled by LLMs and generative AI). For this project we mostly focused on translation of single words and phrases.

3 Background

Lillo-Martin et al. examines the structural properties and linguistic nature of sign languages, drawing comparisons with spoken languages in their paper "One grammar or two?." It addresses some common misconceptions such as the role of iconocity (that the form of a sign is similar to its meaning), and concludes that most non-signers cannot understand a majority of sign language simply based on the signs/icons as previous believed[5]. The paper describes that sign is ultimately very similar to natural languages and thus defines the large scope of the problem of building an ASL Sign Dictionary capable of understanding the context, grammar, and structure and representing it to our users [5].

Authors' Contact Information: Aditya Potluri, apotluri8@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA; May Kalnik, mkalnik3@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA; Smrithi Narayan, snarayan43@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA; Hari Krishna Kumaran, hkumaran3@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA.

ACM XXXX-XXXX/2024/10-ART1
https://doi.org/XXXXXXX.XXXXXXX

The researchers describe the myth of how sign is seen as "pictures in the air" and actually possesses a complex grammatical structure [5]. ASL Signs do not exist in isolation and often rely a sign at the beginning of the sentence to relay time, word order, negation, and more [5]. Any ASL Sign dictionary would have to keep the context that word is in to understand the full meaning. Part of this context includes facial expressions to relay emotion, and thus we found it important teach the sign language using videos that show the person's entire upper half, including their face. [5].

Other researchers have also tested other techniques tackle these contextual challenges. Nunez-Marco et al. provides a survey on the state of sign language translation using computer techniques[9]. The paper breaks down sign language translation into various tasks: detection, identification, segmentation, recognition, translation and production [10]. Within these problems we have seen significant progress in detection, identification, segmentation, and recognition with a variety of glove-based techniques such as one explored by (Das et al., 2016, Gaikwad and Bairagi, 2014, Praveen et al., 2014) and vision based techniques such as Bilge et al. (2022) which all boast very high accuracy in recognition [10]. Our project is more focused on the translation and production aspect and in this regard there is still significant work to be done. Sign language as mentioned previously has a grammar, structure, and meaning as discussed in earlier sources and translation/production is still limited. The paper brings up challenges such as limited datasets and limited techniques to translate natural languages in general[10].

Furthermore, [1] highlights key design considerations and their importance behind creating a user-friendly ASL dictionary. The implementation described uses a recognition model to convert ASL signs to English and explores the best ways to take input and display results. Through testing with Wizard of Oz prototypes, the study evaluated various interface designs and gathered user feedback. The consensus from four participants was that the dictionary's current functionality should remain simple and straightforward, with minimal additional complexity. Although we instead translated from english to ASL, the principle remains and the takeaway aligned with our approach: simplicity leads to a better user experience, and the design should focus on achieving the desired result with as few steps as possible.

Next, let's look at literature relevant to each one of our chosen platforms.

3.1 iOS App

[13] provides a detailed exploration of the development of an American Sign Language (ASL) dictionary built using Apple's QuickTime technology on Macintosh systems. It outlines the challenges encountered during the creation of the application and the innovative solutions developed to overcome them. The key features outlined in the proposal included:

- Video display of ASL signs
- English-to-ASL search functionality
- ASL search based on formation features such as handshape, location, and movement
- Fuzzy search capabilities, allowing users to find signs even when they are unsure of the exact parameters

One of the primary challenges at the time was the speed of retrieval and the limited space for storing video data. The proposed solution involved keeping a small set of signs in memory for quick access, with a larger set stored on disk. However, with modern advancements in memory storage and cloud solutions, this limitation is no longer an issue. For our iOS application, we opted to store no videos directly within the app. Instead, all content is accessed through API calls to a YouTube channel that provides a comprehensive ASL dictionary. This approach ensures fast results and access to a wide array of videos, enhancing both performance and scalability.

Another challenge involved the multiple meanings a single word can have in ASL, depending on context. The proposed solution was to present users with multiple options, including example sentences for each meaning. This remains an essential feature for any ASL dictionary and is a focus for our project as well. To address this, our application returns all relevant ASL signs for the searched word or phrase, along with nearby words, ensuring comprehensive search results. By presenting users with all available options, we eliminate the guessing game and minimize errors in search results. Additionally, we implemented a predictive search feature, allowing users to see suggestions as they type, helping them reach their desired word more quickly and efficiently.

While the original project never came to fruition, we have gone beyond these early challenges by leveraging modern technology to develop a robust and functional ASL dictionary iOS application.

3.2 Android App

The study in [4] emphasizes the necessity of incorporating video signs in sign language dictionaries, as they rely heavily on physical movements for meaning. Video signs not only ensure consistency in learning but also prevent the meaning of signs from varying between individuals. This study focuses on SignDict, an advanced mobile application that leverages technology for rapid and accessible sign language learning.

SignDict stands out for its real-time predictive search, displaying sign options as users type. Additionally, it features a text-to-sign conversion function, which is crucial in bridging English to ASL with minimal human intervention, making the language as universally accessible as possible. The dictionary also standardizes words by returning them in their root form, which further enhances clarity.

The application, developed for Android, utilizes innovative methods to minimize its size, offering "lite" versions to make it accessible in lower-bandwidth areas, particularly in developing countries. Although the study focuses on Indian Sign Language (ISL), its optimization strategies provide valuable insights that can be applied to our own project. Like the multimedia ASL dictionary (MM-DASL), SignDict stores a set of common signs in the phone's memory for quick retrieval. However, we chose not to implement this approach, opting instead for dynamic API calls to YouTube for video retrieval.

Our reasoning was twofold: first, API-based retrieval eliminates the need to store videos locally, allowing us to leverage YouTube's extensive library for sign demonstrations; second, storing some videos locally may prove less useful in the long term. As users become more proficient in sign language, they are less likely to rely on videos of the most common signs, making these stored assets redundant.

SignDict also integrates Tesseract for Optical Character Recognition (OCR) and utilizes YouTube's video library for sign demonstrations. Since our project similarly relies on YouTube as a dataset, incorporating these technologies could enhance the functionality of our application. For now, our product focuses primarily on dictionary features. However, as we expand and scale, integrating computer vision technologies—such as ASL-to-English translation or text recognition from images—could significantly broaden our application's scope in future iterations.

3.3 Chrome Extension

Existing literature details the implementation of a chrome extension (specifically for google meet) that translates sign language into text captions using Machine Learning [7] . The motivation behind using Google Meets is the software's heightened ability to provide personalizations using extensions, especially for educational and corporate settings. [7] describes how a machine learning model was trained to make predictions on ASL specifically using Google's MediaPipe to extract landmark features (from hands) that these predictions were made on. The implemented Chrome extension utilized a JavaScript API to integrate the aforementioned MediaPipe implementations along with MongoDB and Node JS to store the sign information and relay updated information to the extension server in real time respectively.

With this knowledge, we initially aimed to create our chrome extension with machine learning capabilities. However, we soon realized that due to the project's time constraints, that wouldn't be quite possible, especially with such a complex learning task. On the flip side, we were able to implement all of our basic functionalities with JavaScript as in the aforementioned study. Given the proper resources, future work could aim to use tools like MediaPipe in order to approach a machine learning extension.

3.4 Google Docs Add On

One of the questions our team asked ourselves was "why is adding a sign language dictionary more beneficial than just googling how to sign something?" Fajardo et. al. [3] researched the benefits of having what they call a hypertext system in which hyperlinks are embedded around a piece of text that can translate onsite. The authors cite the "Information Foraging Theory (Pirolli and Card, 1999), [which] predicts that a particular hyperlink will be followed when the trade-off between information gained and cost of access is low." [3]. Having this built in plugin will make access to the sign language dictionary easier and therefore more likely to be used. This ability, along with the ability to make queries will be less disorienting for users rather than trying to do a whole bunch of navigation. The Google Docs add on is very similar to the described hypertext system, and therefore proves to be a very viable solution.

This paper also provides an outline for what exactly a user should see when querying or using hypertext. Should it be a human or an animation? Do you see the user's hands or full body? Based on this research and the aforementioned research, we will be continuing to use full upper body videos.

Another source [12] was able to investigate how users interact with the online dictionary. Researcher Mireille Vale conducted user studies to identify the gaps and shortcomings of Online Dictionary of New Zealand Sign Language. For intermediate users, Vale found that they were often looking for sign language grammar and was having difficulty finding the information they needed. This reveals a great challenge for the implementation of a Google Docs Add On. While the user can easily select multiple lines of text, translating each word directly will not provide the user with correct grammar. While we were not able to implement this due to time constraints, this is a great idea for future work.

4 Implementation and Deliverables

As we specified earlier, our deliverables lie entirely in our implementation of an accurate and thorough ASL dictionary that is supported by 4 platforms: an IOS app, an Android app, Chrome Extension, and a Google Docs Add On. Before we dive into the specifics of each implementation, let's discuss commonalities that was purposefully implemented across the platforms.

4.1 ASL Dictionary Interfacing and API Key Generation

These implementations were supported by a Youtube channel (ASL dictionary) that served as our 'dictionary' towards our ASL translations. All of these implementations interfaced an application programming interface (API) call to this channel, using the entered word on the specific platform as a query in order to retrieve the relevant translation videos. To put it simply, a simple request with text would be the same as searching the ASL dictionary with the text and returning the first video found. Generally this video would

provide a natural ASL translation for a given word instead of glossing. It also stored commonly used phrases and idioms.

Each platform had its unique API key (to minimize exceeding API quotas during development and usage). Our team created these API keys using the Google Cloud console with the Youtube Data API v3 service enabled. So overall, our request to the API consists of a channel id (identifying the Youtube channel above), our query encoded, and our generated API key for authorization.

```
// Load the API key from config.json
const config = await fetch(chrome.runtime.getURL('config.json'))
.then(response => response.json())
.catch(error => {
  console.error('Error loading API key:', error);
  throw error; // Rethrow the error so the function doesn't continue
});
const apiKey = config.apiKey
const channelId = "UCACpqsI_FA-gMD2fwil7ZXA";
const url = `${baseUrl}?part=snippet&channelId=${channelId}&q=${encodeURIComponent(query)}&key=${apiKey}`;

const response = await fetch(url);
if (!response.ok) {
  throw new Error(`API request failed with status ${response.status}`);
}
```

Fig. 1. Sample Code Snippet of API calls from Chrome Extension

4.2 Retrieval Logic

When experimenting with our API, we often ran into difficulties with certain translations. As we mentioned above, the simple API request just returned the first video found on the channel, but not necessarily the most appropriate translation. To address this, we filtered the resulting videos with certain rules. Our function first fetches a list of videos with the query as a parameter and filters them to only include those whose titles contain the exact query word. It then sorts the videos so that those whose titles start with the query appear first. It finally returns up to 3 sorted videos following this criteria. We also included logic to not translate definite articles (like "the") as there weren't direct translations.

```
const data = await response.json();
const regexExactMatch = new RegExp(`(?:^|\\W)${query}(?:$|\\W)`, 'i'); // Match whole word with word boundaries
const regexStartsWith = new RegExp(`^${query}(\\s|,|$)`, 'i'); // Match at start, followed by space, comma, or end of string

// Map and filter videos based on exact word match
const matchedVideos = data.items
  .map(item => ({
    title: item.snippet.title,
    description: item.snippet.description || '',
    videoId: item.id.videoId
  }))
  .filter(video => regexExactMatch.test(video.title)); // Only match whole words

// Sort videos to prioritize those that start with the query followed by space/comma
const sortedVideos = matchedVideos
  .sort((a, b) => regexStartsWith.test(b.title) - regexStartsWith.test(a.title))
  .slice(0, 3); // Limit to top 3 results

return sortedVideos;
}
```

Fig. 2. Code Snippet of Retrieval Logic from Chrome Extension

4.3 Style

One key feature we had to implement was not only creating an interface, but also a consistent interface between the platforms so that the user could use all versions of them without having to fully relearn each application. The only problem is that each platform had it's own use cases which inherently would make the front-ends different to accommodate those needs. For example, with iOS and android app, the dictionary is the primary function and takes up 100 percent of the screen and therefore can have more functionality for single words. For the Google Docs and Chrome extension, the webpage/document is the main function, and the ASL dictionary is a side functionality. Also, since a webpage/document is more likely to have blocks of text, it makes more sense to translate phrases here. Our solution was to create two slightly different interfaces based on the use cases. The android and iOS app was created in Figma first because it was slightly more complex. After using the pre-existing iOS app, we were able to identify shortcomings and missing features. Generally our team maintained similar styles for both of our mobile platforms and similar styles for our web platforms due to slight differences in their use cases. We also referenced Nielsons 10 usability heuristics to maximize the efficiency of the our design. These heuristics refer to a set a guidelines UI designers should follow to make sure the interfaces are as usable as possible. To give a few examples of where these heuristics were used [8]:

- User control and freedom: Can easily navigate between previously searched words using arrows, we also provided multiple translations per word so the user can determine whether or not a certain translation "works" for them
- Recognition rather than recall: Predictive search
- Flexibility and Efficiency of use: History allows user to jump around through searched words
- Aesthetic and Minimalist design: We kept the features minimal so as to not overwhelm the user.

4.4 iOS App

The iOS app provided to us served as a foundational base upon which we expanded, refining functionalities and identifying areas for improvement based on our analysis. As mentioned previously, while the app's core functionality was strong—allowing users to type in a word and receive a range of ASL signs from YouTube covering various potential meanings—we identified several opportunities for enhancement. To address these gaps, we implemented improved logic for retrieving relevant words. Instead of relying solely on values automatically retrieved from an API call, our solution ensures that search results align more closely with user intent and context. We also introduced a predictive search feature to enhance usability. After the third character is typed into the search bar, suggestions begin appearing every 0.5 seconds. This reduces query retrieval time and improves efficiency. Additionally, we added a recently searched history feature, which stores the last ten searches. This allows users to revisit previous results without initiating another API call, addressing issues such as accidental navigation or the need to backtrack. To improve the app's overall design and user experience, we made it more intuitive by:

- Adding a home page with clear instructions and functionality descriptions.
- Incorporating a home button to return to the main page after multiple searches.
- Including a back button to revert to the previous state seamlessly.

These enhancements ensure the app is not only functional but also user-friendly, aligning with our goal of creating a more intuitive and efficient experience for users. Just as in the initial app, we used Swift for our iOS implementation due to its seamless integration

with iOS development, strong performance, and user-friendly interface management. Its features allowed us to efficiently implement enhancements like predictive search, search history, and improved navigation.

4.5 Andriod App

Android is the largest mobile platform in the world and developing a working android app greatly expanded the reach of our ASL Dictionary. The Android app was developed to be as visually similar to the IOS app as possible, and was developed from scratch as there was no pre-existing implementation. The app attempts to be as user friendly as possible by implementing a powerful predictive search that uses the most common ten thousand english words and matches the beginning of the search to these words. The app is configured to fetch up to three videos from the API and the user has to scroll through to see the secondary and tertiary options if they exist.

This met the overall goal in that it brought a baseline implementation that fulfills the basic qualities of a dictionary. It falls short of the IOS app in certain aesthetic features such as the home screen and back button since it was started from scratch and the we faced additional challenges with the Android emulator. We hope the application will be further advanced, focusing on maximizing its potential while ensuring it remains intuitive and seamlessly functional for users.

4.6 Google Chrome Extension

Since our team had no experience building a Chrome extension previously, we expected this implementation to be a little difficult, but to our surprise it was more intuitive than we thought. Our codebase for the chrome extension consisted of JavaScript that handled the extension's primary functionalities, and html for styling and displaying the extension and its results. More specifically, in terms of the former, we had two categories of JavaScript files as pictured below; the popup.js file in particular hosts event listeners to be able to translate highlighted text (on a chrome page), clear and search buttons for relevant functions, and methods to not only fetch videos using the aforementioned API but also host logic for selecting videos as described above.
We met the overall goal that we had set out for the Chrome extension in terms of the baseline implementation but like the others we fell short in contextualizing definitions due to our project timeline.

4.7 Google Docs Add-On

The Google Docs extensions were implemented using HTML and a Google Docs Script, which uses JavaScript. This simple tech stack made the extension fairly easy to implement. Because of the nature of Google Docs holding a lot of text, we initially planned to have the Google Docs add-on able to translate entire blocks of highlighted text. This was initially implemented by parsing through each word and translating it directly. After the peer evaluations, we learned more about the complexity of the grammar of translating english to sign language, and that the text couldn't be directly translated as easily as we thought. This was a little out of scope for this project, so we adjusted the add-on to only translate single words or common idioms. We also added functionality for users to search words to better match the google chrome extension and increase the consistency between the platforms. Although we had to adjust the goal of the Google Docs add-on, we were still successful in adding a convenient translator for a user to use without having the navigate to another page.

5 Challenges/Alignment to Class

Our project encountered several challenges. For example, consider homographs—words that share the same spelling but have different meanings and pronunciations (e.g., lead, bass). Without a well-designed dictionary that accounts for multiple meanings or the context of surrounding words, the text-to-ASL translation may result in errors for some of our chosen platform implementations. While we initially intended to use the context of the sentence to extract the intended meaning, we had insufficient time to add the implementation. We instead opted to just show all of the possible definitions of the words regardless of the context. Future work would include implemented grammar into the translations. In addition to this our project taught our group how to plan and execute on building a user-centric application like ASL translation. For example our group heavily utilized the six usability principles such as consistency, feedback, and visibility. These principles ensured a seamless user experience by maintaining uniform design patterns across our different platforms, providing real-time confirmations for user actions by giving slight indications when buttons were pressed, and keeping essential features such as the submit button and text entry front and center. Furthermore, in terms of technical knowledge, we learned plenty about creating, using, and securely storing API keys, Chrome and Google Docs extension development, and android and iOS development.

- Android app: We encountered issues with the emulator that delayed our progress, preventing us from delivering a fully polished product with intuitive buttons. We will continue working on resolving these challenges.
- Chrome Extension: [11] claims that extensions often require ongoing maintenance, as well as monitoring Chrome's guidelines, compliances, and policy changes to ensure they aren't being violated. Furthermore, due to the high volume of extensions and duplicates available on the Web Store, extensions typically face a massive amount of market competition.
- Google Docs: One problem we thought we might have specific to the the Google Docs Add-on would be a potentially very crowded interface. With the other methods (like the apps) the dictionary is the main focus of the application, where as for the add on, the document is the main focus and the dictionary is just a side panel. We solved this by designed different interfaces for the extensions vs the apps. While all 4 implementations maintained consistent and simplistic design, the add-on didn't have as many features (e.g. predictive search, history)

Even after completing our deliverables, our team is still highly enthusiastic about this project. Several members have been closely involved with PopSign, and we are excited to explore how this ASL

dictionary project can integrate with the game app's applications in the future.

6 Results

6.1 iOS App

With the iOS application, we started with a functional app that lacked intuitive design elements. There was no easy way to backtrack, assist with finding words, provide instructions for app use, or present a cohesive design. To improve the user experience, we implemented:

- Intuitive Navigation: Added a home button, a back button, and general information for better usability. (Left)
- Search History: Introduced a recent searches feature to enable backtracking seamlessly. (Middle)
- Predictive Search: Added functionality to suggest words as users type, enhancing search efficiency. (Right)

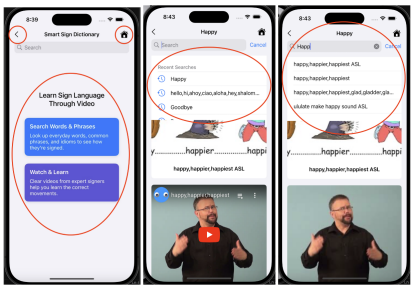

Fig. 3. iOS App interface

6.2 Android App

For android the first iteration was just to display all the basic UI elements on the screen. This includes the webviews/videos and submit button. This allowed for a frame to which I could develop the rest of the apps features. The second iteration was to actually call the API and load a video when the submit button was hit. This created a connection between my app and the dictionary which was critical for the actual function of the app. The third iteration was adding a scrollable web view which could show the multiple results that the web view is capable of showing, this allows for alternate definitions. The fourth iteration is the predictive search which displays the suggestions that match the 10,000 most common words in the english language. This allowed us to reduce the idea to execution for our users and improve our product.

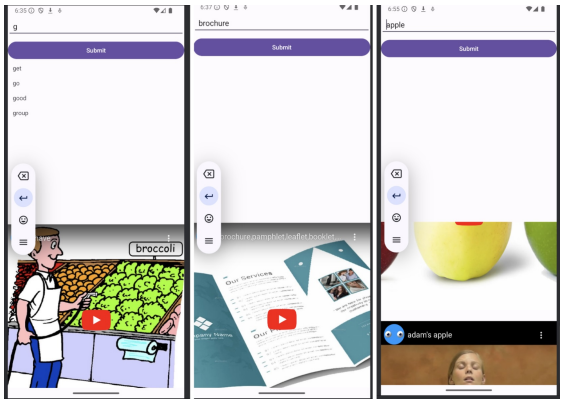

Fig. 4. Android App interface

6.3 Chrome Extension

For the chrome extension, our goal was to first be able to do the basic translation by typing a word into a search box, which is pictured on the left. We felt that the experience would be more smoother and intuitive if the videos began playing automatically and looped, a feature we eventually brought to every platform. Next, we wanted the user to be able to quickly translate any highlightable text on a Chrome page, which is pictured on the right below. Lastly, we implemented logic based on exact word matching to retrieve up to 3 most relevant videos associated with the queries

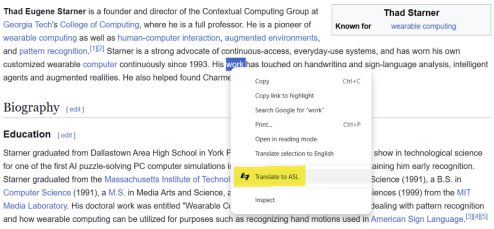

Fig. 5. Google Chrome Extension Interface - search word within webpage

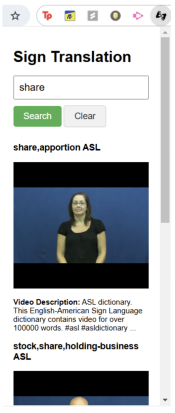

Fig. 6. Google Chrome Extension - Search specific word

6.4 Google Docs Add-On

In the first iteration of the Google Docs Add-On the goal was to get the side panel working. A couple words and their corresponding image translations were hard coded into a dictionary. The code then parsed through all the of selected text and retrieved the image from the dictionary if it existed in it. While this worked as a proof of concept, hard-coding all the words was unrealistic. Later, the YouTube Queries were implemented, making txhis project much more feasible.

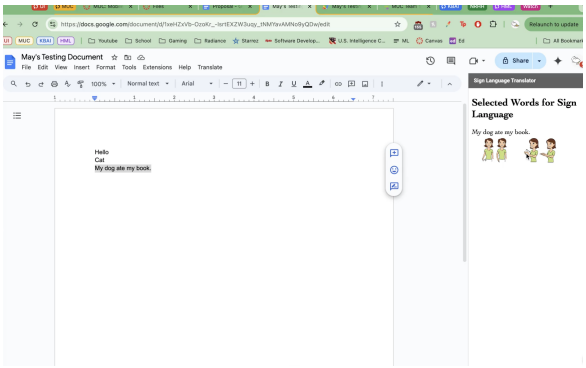

Fig. 7. Google Docs Extension: Version 1

Below is the first feature of the final implementation. The user can select any text from their document. At the menu bar there is a new 'Translate' button that shows two drop down options. If the user selects 'Translate Selected Text', the side bar menu will pop up. As opposed to version one, there will be multiple definitions for each selected word. And the currently parsed word is also displayed for clarity. This works for single words, idioms, and small phrases.

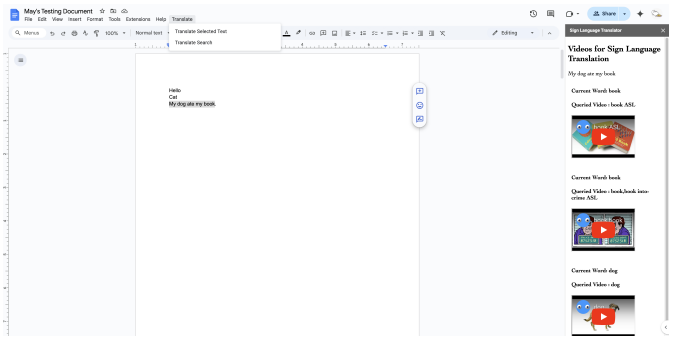

Fig. 8. Google Docs Extension: Search Selected Text

Below is the second feature of the final implementation. The user can open up this side panel and search for any word to see if they could find a translation. This is convenient because the user doesn't have to switch away from the document they are working on to search a word. This function was built for the cases in which the user didn't already have the word in their document to highlight and search. This is very intuitive because it's design both matches the chrome extension and similar online dictionaries.

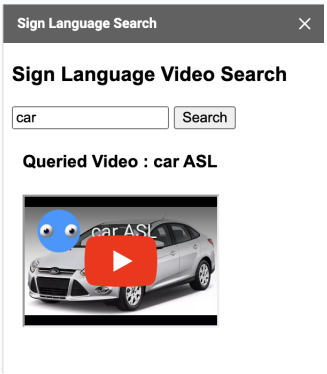

Fig. 9. Google Docs Extension: Search Specific Word

7 Conclusion and Reflection

Our project was generally able to achieve most of our goals, in that we now have a cross platform ASL dictionary that provides natural translations from english to ASL. Our dictionary performs all of the tasks expected of any natural language dictionary in that it can on a word by word basis translate from English to ASL.Certain tasks proved to be far more challenging than we had initially expected such as handling context and grammar. Due to the lack of open source models that can perform the task of taking in english and rendering a natural ASL translation and the difficulty of producing such a model on our own, we were unable to handle context as we suggested in our proposal. Despite these limitations, we believe our work lays a strong foundation for creating a widely accessible reference for individuals unfamiliar with ASL. By bridging the gap between technology and accessibility, we aim to promote a deeper understanding of ASL among a broader audience. This resource has the potential to empower users to engage with the deaf and hard-of-hearing community more effectively. We hope that future work will build upon this foundation to further improve accuracy and usability in ASL learning tools. If future advances in AI allow for a natural translation from english to ASL our app's modularity allows for it to be easily integrated.

8 Acknowledgments

We would like to thank Professor Starner, Sheleah Harris, and the rest of the teaching staff for the opportunity to work on this project, as well as answering our many questions.

9 GitHub

https://github.gatech.edu/hkumaran3/ASL-Dictionary

10 References

References

[1] Matyas Bohacek and Saad Hassan. Sign spotter: Design and initial evaluation of an automatic video-based american sign language dictionary system. In *The 25th International ACM SIGACCESS Conference on Computers and Accessibility*, New York, NY, USA, October 2023. ACM.

[2] Vera Czehmann Dele Zhu and Eleftherios Avramidis. Neural machine translation methods for translating text to sign language glosses. *German Research Center for Artificial Intelligence 1, (2023), 12523–12541*, 2023.

[3] Inmaculada Fajardo, Markel Vigo, and Ladislao Salmerón. Technology for supporting web information search and learning in sign language. *Interact. Comput.*, 21(4):243–256, August 2009.
[4] Jestin Joy, Kannan Balakrishnan, and Sreeraj Madhavankutty. Developing a bilingual mobile dictionary for indian sign language and gathering users experience with SignDict. *Assist. Technol.*, 32(3):153–160, May 2020.
[5] Gajewski J Lillo-Martin DC. One grammar or two? sign languages and the nature of human language. *Wiley Interdiscip Rev Cogn Sci*, 2014.
[6] Shruti Mahajan, Khulood Alkhudaidi, Rachel Boll, Jeanne Reis, and Erin Solovey. Role of technology in increasing representation of deaf individuals in future STEM workplaces. In *2022 Symposium on Human-Computer Interaction for Work*, New York, NY, USA, June 2022. ACM.
[7] Manasee Parulekar Shubhranshu Arya Mrs. Manisha Chattopadhyay, Bhuvana Raisinghani and Varun Bhat. Sign language translation using a chrome extension for google meet.
[8] Jakob Nielsen. 10 usability heuristics for user interface design. *nngroup*, 1994.
[9] Adrián Núñez-Marcos, Olatz Perez-de Viñaspre, and Gorka Labaka. A survey on sign language machine translation. *Expert Syst. Appl.*, 213(118993):118993, March 2023.
[10] Adrián Núñez-Marcos, Olatz Perez-de Viñaspre, and Gorka Labaka. A survey on sign language machine translation. *Expert Syst. Appl.*, 213(118993):118993, March 2023.
[11] Creole Studios. The pros cons of developing chrome extensions in 2024. *Medium*, 2024.
[12] Mireille Vale. A study of the users of an online sign language dictionary. *Create Commons*, 2020.
[13] S Wilcox, J Scheibman, D Wood, D Cokely, and W C Stokoe. Multimedia dictionary of american sign language. In *Proceedings of the first annual ACM conference on Assistive technologies - Assets '94*, New York, New York, USA, 1994. ACM Press.