

Towards a More Inclusive World: Enhanced Augmentative and Alternative Communication For People With Disabilities Using AI and NLP

Zachary Emil, Andrew Robbertz, Richard Valente, Cole Winsor

March 4, 2020

A Major Qualifying Project submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the degree of Bachelor of Science

Authors:

Zachary Emil

Andrew Robbertz

Richard Valente

Cole Winsor

Date:

March 4, 2020

Report Submitted to:

Professor Rodica Neamtu

Worcester Polytechnic Institute

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see <http://www.wpi.edu/academics/ugradstudies/project-learning.html>.

Acknowledgements

This project would not have been possible without the help of so many people. We would like to thank:

- Professor Rodica Neamtu, our MQP advisor, for helping to get our project off the ground, giving insightful feedback, and keeping our team on track.
- Carlos Pereira, CEO of Livox, for making this project possible as well as for providing our team with invaluable feedback and suggestions.
- Andre Camara, professor at UFRPE, for his technical advice and his expertise in Machine Learning and Natural Language Processing.

Abstract

For people with verbal or cognitive impairments, engaging in conversation can be tiresome and time-consuming, limiting their educational, social, and career opportunities. Livox is a pictogram-based alternative communication application that empowers people with a wide range of visual and motor impairments to engage in conversations. This project incorporated a ML and NLP-based classifier to detect specific questions and present the most relevant pictograms to users. Our newly introduced classifier reduced the time and effort required to communicate by 68.5% and 56.4%, respectively compared to the standard application. These results show that our work is a step towards making the world a more inclusive place for those who are nonverbal and have motor skill challenges.

Executive Summary

A recent United Nations report indicates that 15% of the world’s population lives with some form of disability. Many of these people are non-verbal or have verbal and motor challenges. For them, engaging in conversation can require significant effort and become tiresome. Examples include those affected by autism, cerebral palsy, stroke, or cancer. Interlocutors, or conversation partners, additionally require an increased amount of patience due to potentially delayed responses. This leads to a reciprocity gap for people with disabilities when communicating, which in essence refers to a longer wait time for responses in conversation. The United Nations report indicates that the reciprocity gap ultimately leads to challenges for people with verbal and motor disabilities in education, social development, and careers. Therefore, reducing the reciprocity gap is paramount to ensuring inclusiveness of all people into our society.

Augmentative and Alternative Communication (AAC) devices provide non-verbal forms of communication to anyone who has difficulty talking. AACs have proven to be an effective tool to reduce the time and effort required for non-verbal communication. There are many pictogram-based AAC solutions which focus on expressing ideas and constructing sentences using pictograms, including:

- Picture exchange communication systems, that allow users to communicate by handing interlocutors physical pictures
- Recorded speech devices, using dedicated hardware to produce a synthesized voice and speak aloud letters, words, and messages
- Electronic tablet speech applications, running on smart devices such as tablets, smart watches, and portable gaming consoles



(a) Picture Exchange Communication System



(b) Recorded Speech Device

Figure 1: Example AAC Devices

Our research indicated that electronic tablet speech applications are more affordable, more portable, easier to use, and faster to communicate with, compared to other pictogram-based AAC solutions. The largest benefit of electronic tablet speech applications over other AACs is that they are a software based solution. This allows them to have customizable interfaces that make them useful in dealing with a range of mental, visual, and cognitive impairments. Additionally, AAC applications can take a step forward to incorporate machine learning (ML) and natural language processing (NLP) to find new and creative ways to facilitate and enhance communication. In essence, such an approach focuses on predicting the most natural course of a conversation, leading to increased interactivity from users of the application.

Livox is a pictogram-based AAC application that also accommodates a wide range of vision and motor impairments. Livox has a simple yet flexible interface, using a nested-folder structure for

organizing pictograms by category. For example, pictograms related to foods are in the *Food* folder, while those related to family members are in the *People* folder.

Additionally, Livox uniquely incorporates artificial intelligence through a human-centric approach to better facilitate communication for users. Livox analyzes users' past selection and contextual data, including the item, time of use, GPS location, touch duration, and pictogram location on the screen. All these are used to help predict which pictograms a user is likely to need in a given context, and prioritize presenting these items over other available pictograms. Livox also leverages NLP technology to reduce the number of user interactions required to respond to specific, recognizable questions. That is, Livox listens for and detects *yes or no* questions asked in conversation, and facilitates the answer by presenting two full-screen buttons labeled "Yes" and "No".

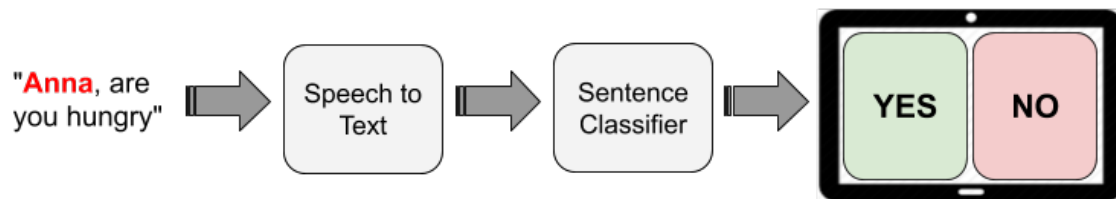


Figure 2: Livox Natural Language Sentence Classifier^{neausi}

The goal of this project was to further reduce the time and effort required to communicate by incorporating an NLP-based list classifier that detects open-ended questions in conversation, and presents relevant response pictograms to users. Our approach listens for *list questions*, which are questions followed by a list of responses, and presents relevant images for each response. We developed a sentence structure to analyze questions based on Amazon Alexa skill's command structure. Our classifier considers three aspects of this structure, including the *wakeword*, the *question phrase*, and the *response phrase*. The *wakeword* indicates that a question is about to be asked, and that the application should record any speech that follows. The *question phrase* is a full question asked, and the *response phrase* is the list of items that should be presented to users of the application.

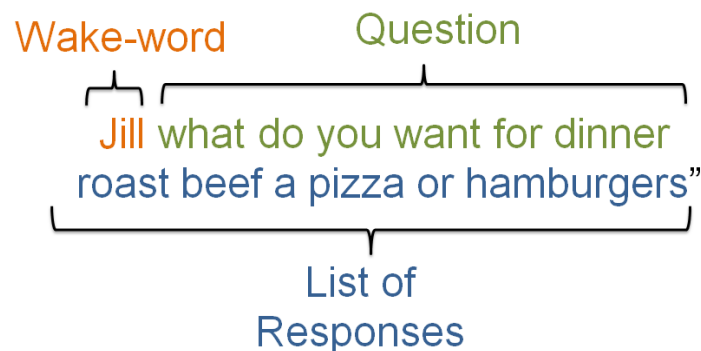


Figure 3: List Question Structure

We identified four objectives that lead to the successful completion of our goal:

1. Classify a question as a *list question*
2. Separate the *response phrase* from the *question phrase*
3. Identify and extract the relevant responses from the *response phrase*
4. Identify the most relevant image for each response

There were two non-functional requirements of our classifier: to function in offline environments and to run on low-end devices. The first is important because Livox is a mobile

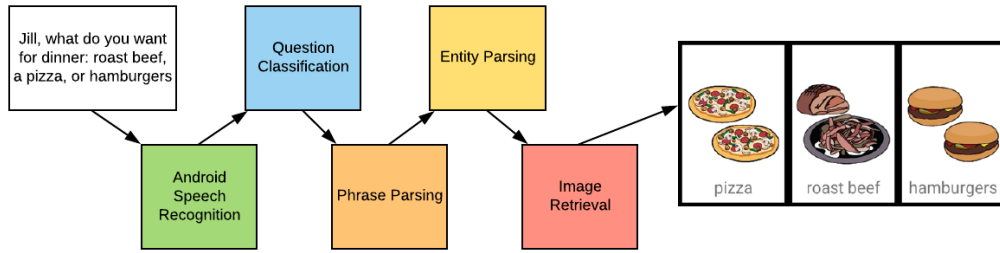


Figure 4: List Classifier Steps^{neausi}

application and supports users both at home or school and while traveling. The second is important because Livox is a low-cost alternative to other AAC solutions and many users rely on older and more budget-oriented devices.

Our approach to solving these problems focused on Agile development and producing frequent deliverables and demonstrations for Livox stakeholders. This Agile approach allowed developers and stakeholders to co-create a successful *list question* classifier across many iterations and through tight collaboration. Early iterations focused on creating an online API hosted by Amazon Web Services (AWS), which was used for rapid development and prototyping of our classifier. Additionally, we created a prototype Android application to interface and test the online API. We later integrated the online solution with the existing Livox application. We created an interface alongside the standard communication board and the *yes or no* classifier interface. Lastly, we ported our online solution to work offline on the Android tablet.

We evaluated our solution using multiple methods, each targeted at measuring the effectiveness of different aspects of our classifier. We tested our classifier for three main purposes, each with one or more methods of evaluation.

1. Data-driven development decisions
2. Measure the accuracy of our classifier’s design
3. Measure the impact our classifier has for users of Livox

To assist our data-driven development decisions, we created a dataset of 30 targeted test questions. We used these questions to ensure that specific functionalities were achievable and to identify weaknesses in our implementation. We created some difficult, non-critical cases as stretch goals for our project. We additionally included cases to test for false positives, questions that our classifier should not parse, in order to identify how our classifier reacted to and recovered from errors in processing.

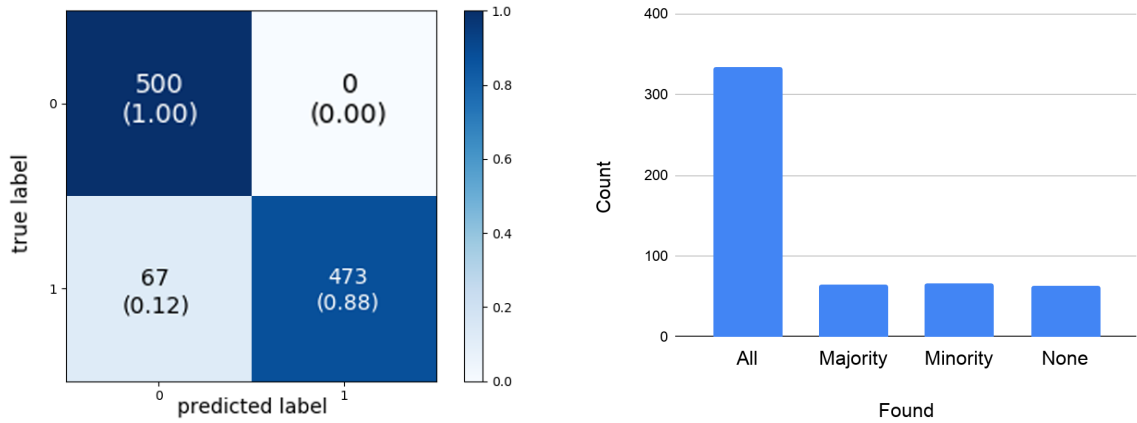
In order to gain an objective understanding of our classifier’s performance, we developed an unbiased, crowdsourced dataset of *list questions*. The dataset includes 530 *list questions* generated by respondents to a Mechanical Turk questionnaire. Each respondent supplied:

1. The full *list question* as it was written by the respondent
2. The list of relevant responses contained within the question (the pictograms ideally presented to users of the application)
3. The conversation topic of the question (classified as one of *activity*, *date/time*, *description*, *locations*, *object/entity*, *organization*, *number*, *people*, or *miscellaneous*)

This data and its subsequent analysis was used to measure how well we achieved each of our objectives. Each full *list question* was sent to our online API which returned:

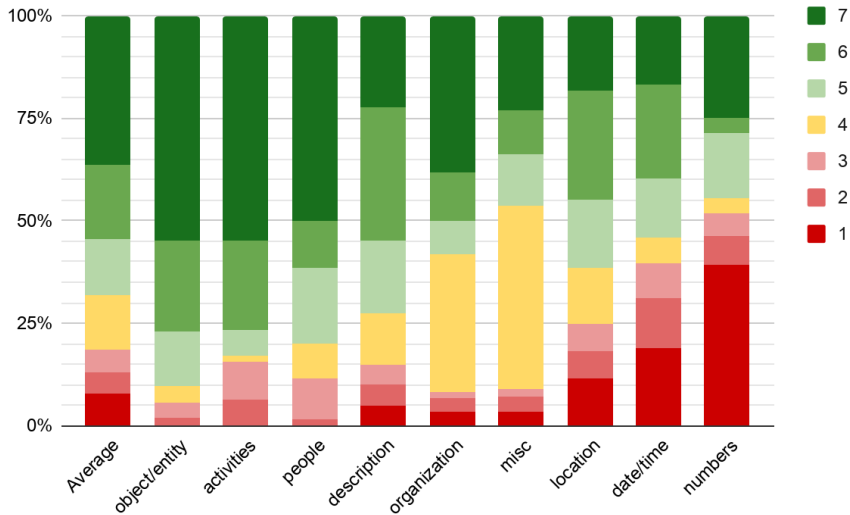
1. Whether the question was identified as a *list question*
2. The split *question phrase* and *response phrase*
3. The list of individual responses that were extracted by our entity parsing service
4. The images that were related to each of the extracted responses

We used the first results to evaluate our classifier’s recognition of *list questions*. We compared the list of extracted responses to those reported in the dataset in order to evaluate the accuracy of our entity parsing service. Lastly, in order to evaluate the relevance of images presented to users, we rated 137 pairings of responses and images extracted from our crowdsourced dataset. We each rated an image based on how well it represented the related response on a scale of one to seven (1-7). We then calculated the mean scores of ratings to identify the overall relevancy of images presented.



(a) Question Classification

(b) Entity Parsing



(c) Image Relevance Scoring

Figure 5: Crowdsourced Testing Results

In order to measure the impact our classifier has on Livox users, we conducted two sets of studies: one focused on quantifying improvements for Livox users, and the other quantifying accuracy for interlocutors. The *user set* was made up of two stages, one using our classifier and one using the standard Livox application, to measure the time and effort required to communicate. We used the results of these stages to quantify the improvements it provides. The *interlocutor set* used three stages

to test our classifier’s ability to detect *list questions*. We used the results from these stages to quantify the ease of use and the time needed to learn the question format expected by our classifier.

The results from our crowdsourced dataset testing and image relevance scoring show that our classifier has an overall accuracy of 94.46% for recognizing *list questions*, 63% for extracting the correct responses, and 68.3% for finding a relevant image. This shows that our classifier reliably activates for *list questions* and consistently presents relevant images for all response options.

The results from our *interlocutor set* show that interlocutors activated our classifier with 88.3% accuracy after minimal explanation of how it functions. This shows that our classifier reliably detects *list questions*, and that the format expected by our classifier is intuitive and easy to learn. The results from our *user set* show that our classifier reduces the time and physical interactions required to communicate, compared to the standard application, by 68.5% and 56.4%, respectively.

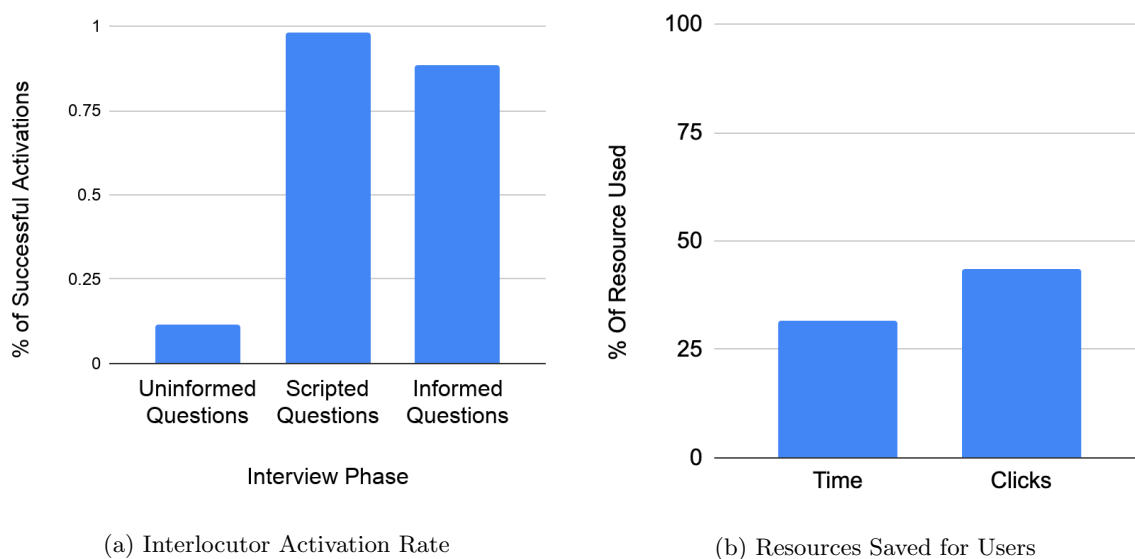


Figure 6: User Study Results

Therefore, we can conclude that our classifier achieves each of the objectives mentioned above and meets our goal of reducing the time and effort required to communicate for users of Livox. We also achieved the non-functional requirements of working in offline environments and on resource-limited hardware. Our project serves as a practical example of how voice-activated functionality can assist people with verbal and motor challenges, and is a step towards ensuring that these challenges can be overcome by applying state of the art technologies in a human-centric implementation.

Contents

1	Introduction	15
2	Background	17
2.1	State of the Art AACs	17
2.1.1	Picture Exchange Communication System	17
2.1.2	Recorded Speech Devices	17
2.1.3	Electronic Tablet Speech Applications	18
2.2	Natural Language Processing	19
2.2.1	Analyzing Sentence Structure	20
2.2.2	Word Stemming	21
2.2.3	Term Frequency - Inverse Document Frequency	21
2.2.4	Word Embedding	21
2.2.5	Machine Learning Classifiers	22
2.2.6	Named Entity Recognition	22
2.3	Livox Application	23
2.3.1	Interactivity Features	23
2.3.2	Livox Portal	24
2.3.3	Innovations in AI/NLP	24
3	Methodology	25
3.1	Classifying List Questions	25
3.2	Separating Question and List Phrases	26
3.3	Extracting Question Responses	26
3.4	Image Classification and Retrieval	28
4	Implementation Strategy	29
4.1	Agile Development	29
4.2	Technology Stack	30
4.2.1	File Organization and Task Management	30
4.2.2	Communication	30
4.2.3	Programming	30
4.2.4	External APIs	31

4.3	Microservice Architecture	31
5	Evaluation	32
5.1	Targeted Testing	32
5.2	Crowdsourced Testing	33
5.3	Image Matching Evaluation	35
5.4	User Study	36
5.4.1	Interlocutor Set	37
5.4.2	User Set	37
6	Iterative Development	39
6.1	Question Classifier	39
6.1.1	Iteration 1 - Hard-Coded Phrase Recognition	39
6.1.2	Iteration 2 - SVM Question Classification Model	39
6.1.3	Iteration 3 - Heuristic Method	40
6.2	Phrase Parser	40
6.2.1	Iteration 1 - Common Question Words Offset	40
6.2.2	Iteration 2 - Improved Question Words	40
6.3	Entity Parser	41
6.3.1	Iteration 1 - Naive Entity Extraction	41
6.3.2	Iteration 2 - Word Embedding Vector Similarity	41
6.3.3	Iteration 3 - Database Derived Vocabulary	41
6.3.4	Iteration 4 - Pre-processing with Word Stemming	42
6.3.5	Iteration 5 - Word Embedding Extended Vocabulary	42
6.4	Image Retrieval	42
6.4.1	Iteration 1 - Google Vision tags Image Retrieval	42
6.4.2	Iteration 2 - Score-Based Image Retrieval	43
6.5	Android Application	43
6.5.1	Prototype Application	43
6.5.2	Livox Integration	44
7	Results	45
7.1	Target Test Results	45
7.2	Crowd-sourced Testing Results	45

7.3	Image Matching Results	48
7.4	User Study Results	48
7.4.1	Interlocutor Interview Results	48
7.4.2	User Interview Results	50
7.4.3	Exploratory Question Results	52
8	Discussion and Related Work	53
8.1	Question Recognition Improvements	53
8.2	Phrase Parsing Improvements	53
8.3	Entity Parsing Improvements	53
8.4	Image Retrieval Improvements	54
8.5	Livox Integration Improvements	54
9	Conclusion	56
	References	57
	Works Cited	57
	Appendix A: Livox Code Analysis	61
	Appendix B: Online Classifier Deployment	64
	Appendix C: Livox Development Environment Setup	67
	Appendix D: IRB Approval Letter	68
	Appendix E: User Study Informed Consent	69
	Appendix F: Recruitment and Screening Forms	71
	Appendix G: User Study Interview Scripts	73
	Appendix H: Mechanical Turk Questionnaires	77

List of Figures

1	Example AAC Devices	5
2	Livox Natural Language Sentence Classifier ^{neausi}	6
3	List Question Structure	6
4	List Classifier Steps ^{neausi}	7
5	Crowdsourced Testing Results	8
6	User Study Results	9
7	Example AAC Devices	18
8	Proloquo2go Application ¹⁸	19
9	Natural Language Processing Pyramid	20
10	Alexa Skill Command Structure ²⁵	20
11	Simple Vector Word Representations	22
12	Livox Natural Language Sentence Classifier ^{neausi}	24
13	List Classifier Steps ^{neausi}	25
14	List Question Structure	26
15	Stop-Word Removal	27
16	Tokenizing Responses	27
17	Response Stemming	27
18	Google Vision API Generated Labels For an Image	28
19	Modular Model Evaluation	33
20	Mechanical Turk Batch Results	34
21	User Study Results	35
22	Image Rating Topic Distribution	36
23	Average Scored Examples	37
24	Returned drain labeled as paw by Google Vision vs Livox tagged paw	43
25	Targeted Testing Results	45
26	Question Classification Confusion Matrix	46
27	Phrase and Entity Parser Results Examples	46
28	Phrase and Entity Parsing	47
29	Image Relevance Results	49
30	Interlocutor Set Results	50

31	User Study Results	51
32	Resources Used	52

1 Introduction

A 2018 United Nations report indicates that 15% of the world’s population lives with some form of disability¹. A large portion of these people have difficulty speaking, and engaging in conversation can require significant effort and become tiresome. Interlocutors, or conversation partners, additionally require an increased amount of patience due to potentially delayed responses. This can create a reciprocity gap for people with disabilities when communicating, and limit their ability to integrate as a member of society. The United Nations report indicates that the reciprocity gap creates challenges for these individuals in their education, social development, and careers. In the US alone, it is estimated that 1 in 345 children have cerebral palsy, one of many disabilities that can affect speech². A recent UNICEF study indicates that almost 50% of children worldwide with disabilities do not attend a formal school, and nearly 85% have never received a formal education. Even for those who do, there is evidence that traditional schooling methods are not suitable to meet their needs³. The reciprocity gap for the verbally impaired must be reduced as much as possible in order to help them in their education, social development, and careers.

Augmentative and Alternative Communication (AAC) devices have emerged as an effective tool to reduce the time to communicate for individuals with disabilities⁴. However, they are not capable of completely eliminating the reciprocity gap. Natural language processing (NLP) and machine learning (ML) can be applied to communication between users and interlocutors using a more human centric approach. Such an approach focuses on predicting the most natural course of a conversation to allow quick responses.

Livox is a unique, pictogram-based AAC application to facilitate communication through a simple, yet highly customizable interface that also accommodates a wide range of vision and motor impairments. Livox has made great strides to implement human-centric ML and NLP tools into their application, such as an NLP-based binary *yes or no* question classifier. This empowers interlocutors to ask *yes or no* questions, which triggers the application to display the appropriate “YES” and “NO” options for users to select in response. This and other features empower users to quickly and easily communicate with interlocutors.

The primary goal of this project was to reduce the time and effort required to communicate for people with disabilities by incorporating an NLP-based multi-label classifier into Livox’s machine learning infrastructure. Our approach listens for *list questions*, questions followed by a list of responses, and presents relevant images for each response to users. We identified four objectives that lead to the successful completion of our goal:

1. Classify a question as a *list question*
2. Separate the *response phrase* from the *question phrase*
3. Identify and extract the relevant responses from the *response phrase*
4. Identify the most relevant image for each response

In order to achieve each of these objectives, we utilized an iterative process, focused on Agile development and producing frequent deliverables and demonstrations for Livox stakeholders. In order to guide or iterative improvements, we identified four design goals to focus our efforts and move towards the most effective solution within Livox. Livox is a cost-effective AAC solution and is used in remote and less advantaged parts of the world. Therefore we aimed for the classifier to:

- Have high accuracy for analyzing common speech,
- Encapsulate natural conversation to facilitate ease of use,
- Run in offline environments independent of internet connection,
- Run efficiently on older or less powerful devices.

When faced with design and implementation decisions, we considered each of these factors in our resolution. In the following sections, we outline how NLP and ML classifiers can be used to facilitate communication through AAC devices, as well as outline the specific steps we took to achieve each of our objectives and design goals.

2 Background

Augmentative and alternative communication (AAC) systems are necessary tools for many people with disabilities to interact and develop socially. There are several categories of AAC systems, such as picture exchange communication systems, recorded speech devices, and electronic tablet speech applications. Each approach comes with its own set of trade offs, which have to be weighed on a case-by-case basis, based on the needs and preferences of the user. Natural language processing and machine learning can be applied to electronic tablet speech applications to further reduce the reciprocity gap and physical effort required by the user. In this section we outline the state-of-the-art AACs within each category, and identify strategies for how NLP and ML can be used to understand and facilitate communication.

2.1 State of the Art AACs

The goal of AAC systems is to provide non-verbal forms of communication to anyone who has difficulty talking⁴. These people can include those affected by autism, cerebral palsy, stroke, cancer, or other conditions that affect verbal communication. The use of AAC devices can range from short term, such as during recovery, to long term, even lifelong⁴. The research community has collectively created a plethora of AAC systems, with some focusing on the construction of sentences using subsets of selected words, and others focusing on communication through pictograms. This paper strictly focuses on the later subset of devices. Pictogram-based AAC systems’ user experience can be analysed by their affordability, portability, ease-of-use, and time-to-communicate. Due to AAC’s usefulness to a variety of conditions, these criteria must serve a variety of user’s individual cognitive, visual, and fine motor skills⁵. The major forms of pictogram-based AAC systems were analyzed using these criteria, including:

- Picture Exchange Communication
- Recorded Speech Devices
- Electronic Speech Tablet Applications

Picture exchange communication systems are affordable and allow an extensive vocabulary, but lack portability and take lots of time to communicate. Recorded speech devices are more portable and quicker to communicate with, but are more expensive and have smaller vocabularies. Electronic speech tablet applications provide a balance between these different concerns. In addition, electronic speech tablet applications take advantage of software updates to become more customizable and make better use of cutting edge technology.

2.1.1 Picture Exchange Communication System

Picture exchange communication (PEC), as seen in Figure 7a, is one of the simplest AAC systems⁶, allowing individuals to communicate with interlocutors by handing them physical pictograms, pictures expressing thoughts, needs, or desires. Many pictograms are needed as each picture is only intended to communicate one concept. Due to this, transportation can be difficult and the system may not be easily available outside of environments such as the home or school. The time-to-communicate can be long, as the pictogram must be found and passed to the interlocutor. Despite this, PEC systems tend to be very affordable, having many free online resources⁷, as well as easy to use, as the teaching process is well documented⁸. This system is customizable for different visual abilities and cognitive levels, however, users must have good motor skills because it requires handling multiple cards at a time.

2.1.2 Recorded Speech Devices

Recorded speech devices (RSDs) “produce electronic voice output, allowing the individual to communicate” by using a synthesized voice, to speak aloud letters, words, and messages¹¹. One



(a) PEC⁹



(b) GoTalk Express 32¹⁰

Figure 7: Example AAC Devices

pictogram-based RSD, the GoTalk Express 32, can be seen in Figure 7b¹⁰. This RSD offers 32 messages at a time, allowing for simple communication. In addition, a background card can be exchanged with up to 4 other pre-set cards to change all 32 messages that can be selected. GoTalk Express 32 is designed for portability, including easy transport handles and a long battery life. Most pictogram-based RSDs provide this same functionality, but support different amounts of messages and cards.

Pictogram-based RSDs are simple to use as they present a limited number of options in static positions. The statically-positioned choices make RSDs easy to learn and this can speed up communication. However, if the current card doesn't include the desired option, swapping cards to get the needed message can greatly increase the time to communicate. For example, if a user wanted to ask for lunch but the card they were using was about animals, as is the case in Figure 7a, then they would need to find and swap to a card with food options. Unfortunately, RSD systems can be expensive, ranging from about \$165¹² up to \$700¹⁰. These systems lack customizability for different cognitive, visual, and motor abilities, but the variety and target user base of companies making RSDs means that most users will be able to find a suitable device.

2.1.3 Electronic Tablet Speech Applications

Electronic tablet speech applications are a specialized type of pictogram-based RSDs which run on smart devices such as tablets, smart watches, portable gaming consoles, and others. They are developed both for academic purposes, as is the case with Sc@ut¹³ and IMLIS¹⁴, as well as in industry, as is the case with Proloquo2go¹⁵, Snap + Core First¹⁶, and TalkTablet¹⁷. These applications provide the same picture to speech functionality as RSDs, however the use of smart devices give several advantages. This approach allows dynamically exchanging pictograms, letting the application display pictograms in logical groups and allowing users to quickly navigate to relevant options. AAC applications also allow users to create their own pictogram content. In addition, application visuals can be fully customized to the needs of the user. Most smart-device applications use a nested-folder structure for organizing pictograms by category.

AAC applications allow the creation of modern features which allow new approaches to AACs. Proloquo2go is an AAC application using a fine grained approach to language, allowing them to focus on the growth and development of an individual's vocabulary¹⁸. This application adds extra features to those above to achieve this focus. The application uses a tiered learning system in which users start with a core vocabulary, words which are used most frequently. As the user develops stronger cognitive abilities and becomes more comfortable with their core vocabulary, new words and grammatical options are added. As seen in Figure 8, the core vocabulary is displayed on the main page, while "fringe words", which are used less frequently, are organized into folders for different topics on the bottom two rows of the screen. This application also provides users with the ability to use any verb

tense, accessible by holding down a tile with a shaded top right corner.

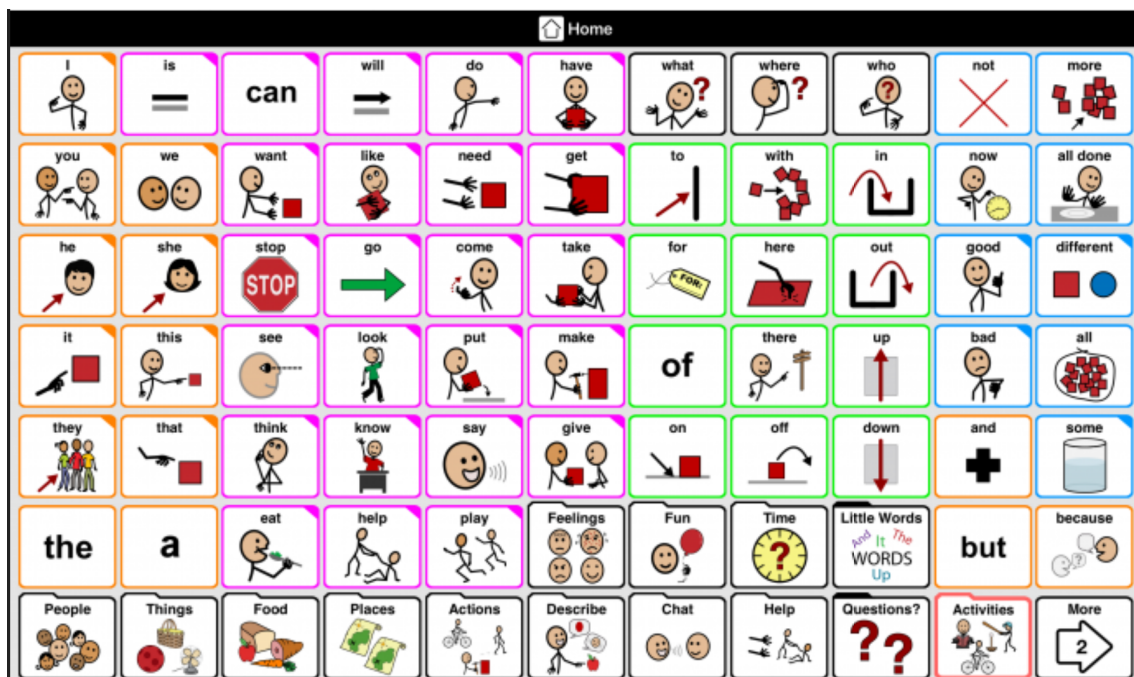


Figure 8: Proloquo2go Application¹⁸

AAC applications tend to be less expensive than other RSDs, as Proloquo2go is only \$250¹⁵. In addition, the portable design of most tablets and other smart devices make these applications easy to transport. However, smart-devices require frequent charging or constant power, decreasing portability of the AAC solution. AAC applications provide simple and concise controls for finding desired pictograms which can decrease the time to communicate for large vocabulary. However, the time to communicate can still be longer than simpler RSDs, especially if the tiles are not always arranged in the same order or if the desired pictogram is on a different page, requiring more physical interactions from the user^{neausi}.

The largest benefit of electronic tablet speech applications over other AACs is that they are a software based solution. This allows them to have customizable UI's to make them accessible to a range of mental, visual, and cognitive disabilities. AAC applications can be continually developed, with new features immediately available to users through software updates. Finally, AAC applications can take advantage of cutting edge technology, such as machine learning and natural language processing, to create new ways for AAC's to help their users.

2.2 Natural Language Processing

Natural Language Processing (NLP) is at the intersection of the fields of computational linguistics and artificial intelligence¹⁹. It encompasses the algorithms which allow computers to “read, decipher, understand, and make sense of the human languages²⁰²¹”.

In order to understand and extract information from pieces of text, NLP algorithms typically do analysis in four linguistic stages: morphology, syntax, semantics, and pragmatics, as seen in *Figure 4*²². Morphology is analysis of the words which make up a sentence. This stage includes the process of lemmatization, which normalizes words by removing affixes to reduce them to a base word²³. The syntax analysis stage focuses on the relationships between words²² and includes finding parts-of speech and building a dependency tree between words²³²⁴. Semantic analysis refers to understanding the sentence as a whole. NLP tasks at this stage include named entity recognition and sentiment analysis²². Finally, pragmatic analysis uses surrounding sentences and phrases to identify the context of a particular sentence and to identify the meaning of a text as a whole²¹. The stages used, level of

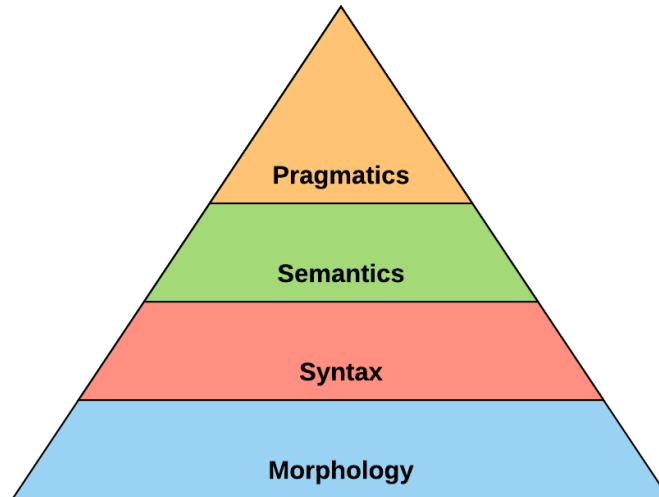


Figure 9: Natural Language Processing Pyramid

analysis, and algorithms used are entirely dependent on how NLP is being applied to a given task.

Specific techniques can be employed at different stages of NLP to create a larger pipeline of processes. Analyzing sentence structure can be used at the semantic stage. Word stemming can be used at the morphology stage. Word Embedding and term-frequency-inverse document frequency (TF-IDF) can be used at the syntactic stage. Named Entity Recognition (NER) can be used in both the syntax and semantics stages. Machine learning classifiers are used at many stages in NLP, and can be used to achieve powerful pragmatic analysis.

2.2.1 Analyzing Sentence Structure

One strategy used in syntactic analysis is to define specific, expected sentence structures to be recognized by the rest of the processing. An ideal sentence structure must fit two requirements: it must be fluid enough to encompass natural conversation and it must be structured enough to make it easy to identify and parse. These two goals are at odds, where one requires a more fluid structure, and the other requires a more rigid structure.



Figure 10: Alexa Skill Command Structure²⁵

Voice activated devices, such as the Amazon Echo²⁶ and Google Home²⁷, allow the use of voice commands to interact with provided services. Amazon allows users to create custom commands, coined Alexa Skills, which can be deployed onto any supporting device²⁸. Each Alexa Skills command is broken into several pieces, seen in Figure 10: the wake word, the launch phrase, the invocation phrase, and the utterance²⁵. Sentence segmentation breaks the sentence down into phrases, so different information can be extracted from each phrase. The wake word precedes any command, and acts to wake the system up. For example, Amazon uses the name “Alexa” as a wake word. The launch phrase is what identifies the action the user is requesting. In an Alexa Skill this is a verb such as “launch”, “ask”, or “open”. The invocation gives context to the action by identifying the intent of the action. Alexa Skills use the skill name as the invocation which identifies what the user intends to interact with.

Together the launch phrase and invocation define what the command is. Finally, the utterance contains the details of what that command should do. Below is an example of how a sample sentence is broken up.

2.2.2 Word Stemming

Word Stemming is used to create more lightweight models that can recognize different forms of the same word. It is a heuristic based method that attempts to remove affixes from either end of a word²⁹. For example the words “person”, “persons”, “personality” or “personalities” would all be interpreted as “person”. This method has its flaws but generally works well. A more robust strategy is to use word lemmatization. Instead of heuristic based, lemmatization is tree based, utilizing formal root words that construct a word. Although word lemmatization is more accurate, it has higher computational and storage requirements²⁹.

2.2.3 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) is a method of scoring a large number of documents when searching for a query. It can be used by search engines to rank web-pages for specific searches. Term Frequency scoring gives a higher score to documents the more a term or a word in the query appears. A problem arises when scoring only by Term Frequency, where a term may appear in a longer document more than in a shorter one, causing the longer one to score higher.

Inverse Document Frequency addresses the relevance problem in Term Frequency by accounting for the number of documents in which the term is found, determining how common and useful it is. Inverse Document Frequency for a term is defined as $idf = \log(\frac{N}{df})$, where “df” is document frequency, the total number of documents in which the term is found, and “N” is the total number of documents gathered. A query term will be weighted less the more often it appears in multiple documents, but will be weighted with higher importance if the term appears in very few documents. The Term Frequency score is adjusted using the Inverse Document Frequency simply by multiplying the terms together. Therefore, an entire document can be searched using all terms in the query: $Score(q, d) = \sum_{t \in q} tf_{t,d} \times idf_t$. The documents with the highest score are selected as the most relevant to the query text^{29,30}.

2.2.4 Word Embedding

Word embedding is the process of converting textual words into numerical vectors, sometimes composed of 300-400 numerical dimensions. This can be achieved using a number of ML libraries, including Word2Vec, which is trained on a large text dataset and converts a dictionary of words into vectors. Word2Vec also considers in part the relative placement of words around their neighbors in the text³¹. These vectors can be compared for similarity within the vector space.

For example take the four words: “lunch”, “hamburger”, “tree”, and “forest” . The words “lunch” and “hamburger” would be closer in the vector space than “lunch” and “tree”. As seen in Figure 11, this allows for the use of a domain specifier to reduce ambiguity for compound nouns. For example, distinguishing between “hot” and “dog” versus “hot dog” would be much easier knowing the domain is “lunch”. Word embedding can also be used to extend a vocabulary to recognize synonyms of a word. For example, if the label of an image is “pajamas” word embedding can be used to find the closest n words to the given phrase, allowing for recognition of slang phrases such as “jammies” to be connected to “pajamas”. Overall word embedding allows users interacting with our protocol to use more natural conversation and a richer vocabulary.

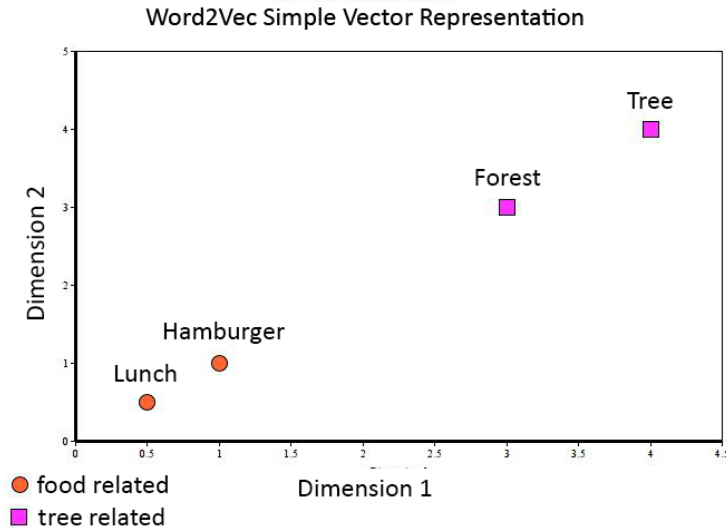


Figure 11: Simple Vector Word Representations

2.2.5 Machine Learning Classifiers

Machine learning is a branch of artificial intelligence that focuses on systems that can learn a function without its behavior being explicitly programmed³². Machine learning is used to solve problems with high input complexity where it is impractical to handle problems on a case by case basis. For example, machine learning is used in Google’s Android Speech Recognition (ASR) library to achieve speech to text capability in over 120 different languages³³.

Convolutional Neural Networks (CNNs) are a type of neural network designed to leverage the complexity of large input vectors. Typically they are used for images but have also been found to be effective within NLP tasks. CNNs aim to mimic the brain’s visual system by implementing many convolutional layers that are each targeted at recognizing specific features. For example, in image processing networks, early convolutional layers may focus on simple features such as lines and curves, while deeper layers may use lower level features to identify more complex features such as a nose or mouth.

CNNs can also be applied to understanding language. Sentences can be represented numerically within a vector space using TF-IDF or word embedding, and used as input to an NLP CNN³⁴. Larger pieces of text can be parsed and vectorized to create sentence matrices. These matrices can be convolved to classify sentence categorizations and to gain semantic and pragmatic information. Semantic and pragmatic analysis can then be used to identify the themes and topics of a larger text.

Long Short-Term Memory Networks (LSTMs) are a type of network commonly used for time-series data. Similar to a Recurrent Neural Network (RNN), they allow for the maintenance of an internal state that is persistent from previous data³⁴. LSTMs improve upon RNNs with the addition of a forget gate that allows for more continual usage over long sequences of data³⁵. LSTMs are designed to react to continuous streams of data used in cases such as autonomous driving.

2.2.6 Named Entity Recognition

Named entity recognition (NER) is the process of identifying expressions that refer to people, places, organizations and companies³⁶. NER adds semantic information to the sentence and can support pragmatic text analysis. For example, “John went to California to visit Apple headquarters” could classify “John” as a person entity, “California” as a place entity, and “Apple” as an organization entity. In this example, NER resolves the ambiguity in the word “Apple” meaning either a fruit or the company. By tagging John and California, NER additionally identifies that “a person went to a place”

rather than “a noun went to a noun”.

There are several available NER libraries, such as Stanford NLP³⁷, spaCy³⁸, NLTK³⁹, and GATE⁴⁰. Stanford NLP is a powerful Java library that includes a variety of pretrained models, as well as tools for training custom models⁴¹. This library uses a linear chain conditional random field (CRF) sequence model³⁷. CRF models are a type of discriminative classifier, which directly estimate the conditional probability between the input and output classes⁴². CRF models are commonly used in NLP tasks because they take context from neighboring samples when classifying each individual sample³¹.

2.3 Livox Application

Livox is an electronic tablet speech application, available for Android devices, designed to accommodate users with vision, motor, speech, and other impairments. Livox is designed to have a simple yet flexible interface, making it quick and easy for caregivers to setup and use the communication board. The application is highly customizable, allowing for multiple users to coexist on the same device, each with their own specific profile with interface settings and collections of pictograms. Users can download, create, and share additional content through a web interface. Lastly, Livox uniquely incorporates artificial intelligence through a human-centric approach to better facilitate communication for users. Although some features within the application might seem trivial, and are provided by other existing AACs, the holistic collection of features included in Livox provide an all-in-one solution that can be adapted to almost any user.

2.3.1 Interactivity Features

Livox provides a unique set of tools for caregivers to setup and maintain multiple devices, each with their own set of user profiles that focus on the usability and simplicity for users. The application enables seamless and instant transition between multiple profiles on a single device, allowing it to be used for multiple users at a time. Caregivers can create profiles with basic information about the user’s impairments, which will initially specify their interactivity options.

In order to better facilitate communication, users or caregivers can change the following configurations at any time:

- Increase or decrease the number of rows and columns of items, effectively increasing or decreasing the size of items on the screen.
- Present normal pictograms, black and white images, or high-contrast images to accommodate vision impairments.
- Display pictograms at full screen size after selection, so users can validate that the correct item was selected.
- Set a customized click interval, which forces the application to wait a specified amount of time before recognizing successive clicks.
- Select custom navigation options including fixed or hidden buttons and gestures through swiping.
- Return to the main screen after a specified time interval so users do not have to repeatedly select the back button.
- Maintain pictograms’ relative position on the screen, even when some pictograms are hidden, so that the same items are in consistent locations.

Livox also includes automated imprecise touch adjustment (IntelliTouch), which tracks how many fingers are touching the screen, if the hand was dragged across the screen, the duration of the touch, and other factors to correct for the imperfect touch of users with motor-skill impairments. This

collection of features sets Livox apart from its competitors, making it one of the most adaptive and accommodating AAC solutions.

2.3.2 Livox Portal

The Livox Portal provides a place for device managers to configure and gather statistics on their devices. Through this portal, managers can gather information about the number of unique users on a device, the version of software they are using, their language preferences, and their geographic regions. Managers can additionally, if applicable to the user, monitor how users are making improvements in their education.

The Livox store provides a place for users to create, upload, share, and download custom-made pictogram tiles, all curated by Livox. Users can create and search for labels on specific sets of pictograms, as well as rate content to find the most popular. By involving users in content creation, there is no upper limit on the kinds of content that can become available for Livox.

2.3.3 Innovations in AI/NLP

Livox takes a unique approach to using artificial intelligence by focusing on a person’s context to facilitate communication. Livox analyzes user’s past selection data, including the item, time of use, touch duration, GPS location, and pictogram location on the screen^{neausi}. Using this information, machine learning algorithms are used to predict what items a user is most likely to select within a given context. For example, if Livox detects the device at a restaurant and the time is around noon, it will prioritize food items often selected at this time and location. If tiles must be split onto multiple pages, it will display prioritized items on an earlier page, and normally available items on successive pages. In this case, it will likely prioritize tiles “I want...”, “to eat...”, and foods often selected for lunch.

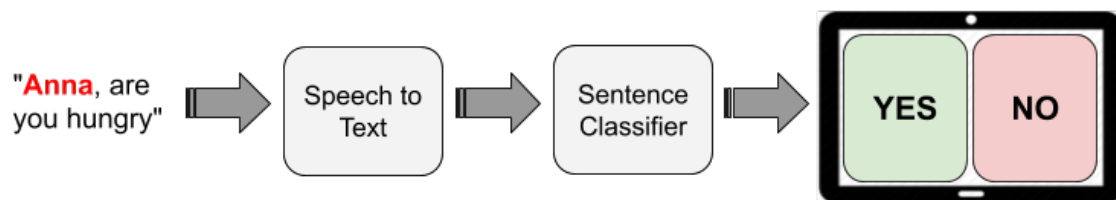


Figure 12: Livox Natural Language Sentence Classifier^{neausi}

Livox leverages NLP technology to reduce the number of user interactions required to respond to specific, recognizable questions asked by interlocutors. The application continuously listens for the specified wakeword and, once recognized, activates the application’s question classification algorithms. As seen in Figure 12, these classifiers currently recognize *yes or no* questions and, if detected, will replace the current screen with two full-screen buttons labeled “YES” and “NO”.

3 Methodology

The primary goal of this project was to expand the use of Livox’s context-aware machine learning and natural language processing to reduce the time and effort required to communicate for people with disabilities. These technologies can be used to develop a suite of human-centric tools to facilitate bidirectional, natural conversation, however, this paper focuses on the implementation of one natural conversation tool.

Currently, users are required to manually navigate many menus to reach the desired tile when asked a question. Our tool enables interlocutors to ask questions followed by a list of possible responses, displaying several tiles with the most relevant images for each response. With this Livox natural conversational tool, users will be able to immediately select their answer from a short list of items presented.

This feature was divided into four objectives in order to simplify the development of our natural conversation tool:

1. Classify a question as a *list question*
2. Separate the *response phrase* from the *question phrase*
3. Identify and extract the relevant responses from the *response phrase*
4. Identify the most relevant image for each response

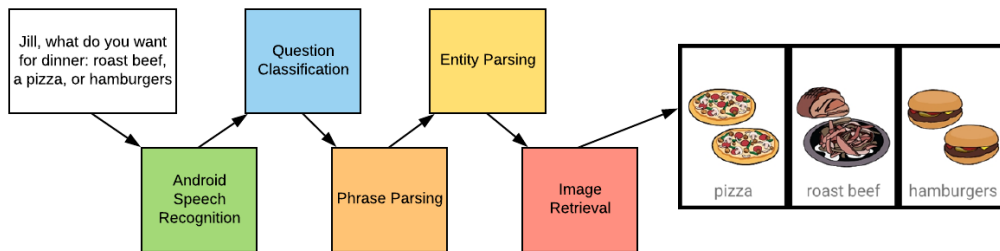


Figure 13: List Classifier Steps^{neausi}

There were a number of restrictions on the project including both functional and non-functional requirements. Functional requirements included encapsulating common and slang speech, and encapsulating phraseology used in natural conversation. Non-functional constraints on our implementation included running on low-end mobile devices and running without an internet connection.

3.1 Classifying List Questions

In order to simplify classification of recorded sentences, we created a predefined sentence structure to be recognized. This structure was designed to be rigid enough to allow for question classification, yet generic enough to project a natural feeling when being used by interlocutors. We modeled our approach after Amazon’s command structure in their Alexa Skills service. This included a wakeword, a combined launch and invocation phrase, and the utterance phrase.

The first aspect of the command structure is the wakeword, used to trigger all voice interactions within the application. In the example *list question* in Figure 14 Livox already recognizes the wakeword “Jill”. The wakeword in Livox can be defined and changed in the application’s configuration, but it is most commonly the user’s name. The Livox wakeword was used in our *list question* classifier without any addition or modification to the codebase.

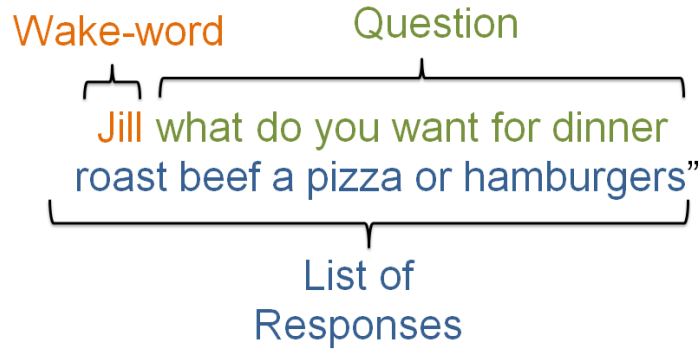


Figure 14: List Question Structure

We combined Amazon’s concept of the invocation and launch phrases to define our *question phrase*. In the example in 14, the *question phrase* is “what do you want for dinner.” It should be noted that the *question phrase* does not need to be a complete statement. For example the phrase “Jill, would you like pizza or hamburgers?” is also acceptable. Finally, we considered Amazon’s concept of the utterance phrase to be the list of possible responses succeeding the *question phrase*. In the example in 14, the *response phrase* is “roast beef, a pizza, or hamburgers.” This sentence structure enabled the processing of questions within our NLP solution.

In accordance with our sentence structure, a recorded question must contain two parts after the wakeword; the *question phrase* and the *response phrase*. First, the *question phrase* is identified using a recognizable question initiator, such as “do,” “what,” or “are.” Second, the *response phrase* is identified by a coordinating conjunction connecting two or more similar nouns in the list of entities. The word “or” is an example of this in Figure 14. Our classifier detects *list questions* based on these two defining factors.

3.2 Separating Question and List Phrases

In order to determine where the *question phrase* ends and the *response phrase* begins, we developed a heuristic solution that relies on detecting two key factors of the recorded text; the *question word* and a predefined offset number of words to use when looking for the *response phrase*. Common question sentences follow fairly consistent patterns, such as:

- “**What** would you like **to** eat || pizza, pasta or steak?”
- “**What** would you like **for** breakfast || eggs, bacon or ham?”

Thirteen common *question words* such as “who”, “what”, “when”, and “where” are first searched for, then secondary closure terms such as “for” or “to” are found in the question. The secondary terms have offset values assigned to them, usually between 0 and 2. In the case that no secondary term is found, we assumed an offset of 4 words until the beginning of the *response phrase*. The *response phrase* is assumed to start after the offset value’s number of words from the secondary term. Since there are many cases where one offset value is not always correct, conservative values were used preventing important content from being removed from the *response phrase*. We believed it was better to have potentially extraneous entities on-screen, rather than missing one or more entities from the list.

3.3 Extracting Question Responses

After separating the *question* and *response phrases*, our classifier extracts all relevant responses to the question. For example, given the separated *response phrase* from Figure 14 “roast beef, a pizza,

or hamburgers,” our classifier extracts the entities, “roast beef,” “pizza,” and “hamburger.” We are able to achieve this through three steps of analysis, including:

1. Removal of stopwords
2. Tokenizing responses to find the most relevant combinations of bi-grams and trigrams
3. Word stemming

Stop-words are the most common words used in natural language, and include words such as “the,” “at,” “for,” and “from,” along with countless others. For example, in the phrase in Figure 15 the removal of stop-words is necessary so that our classifier does not include words such as “a” and “or” in the list of responses presented to the user. This is easily done through a dictionary lookup of common stop-words. Many NLP libraries, such as NLTK for Python, have built-in dictionaries for such words, with the ability to define custom dictionaries if desired. Our solution uses a default dictionary to transform the phrase “roast beef, a pizza, or hamburgers” to be “roast beef, pizza, hamburgers.”

roast beef ~~a~~ pizza ~~or~~ hamburgers?

Figure 15: Stop-Word Removal

In order to recognize compound word entities, such as “roast beef,” our classifier recognizes which pairings of words in the list of responses should be grouped together. In order to achieve this we created a dictionary of common compound word responses that are known to have images associated with them.

Once the *response phrase* is cleaned, the list of responses is tokenized to extract a superset containing each individual word, as well as all two-word and three-word combinations that include surrounding words. We then search this superset of possible entities for known compound words contained in our vocabulary. In this stage we give precedence to recognizing longer compound word entities, and search for single-word responses last. For example, the cleaned phrase in Figure 16 “roast beef, pizza, hamburgers” has subsets with “roast” and “beef”, and others with “roast beef”. The latter are always chosen over the smaller combinations of words, in each case, just a single word.

roast-beef pizza hamburgers

Figure 16: Tokenizing Responses

Additionally, we used word embedding to augment and extend the vocabulary of entities in our dictionary. Using a word embedding model generated by Google, we added support for ten synonyms for each entity in the vocabulary. For example, the word “pajamas” in the vocabulary was supplemented with words such as “pjs” or “jammies” from the word embedding model. This is used when the extracted entity does not exactly match the Google image labels, and greatly increases the volume and diversity of our classifier’s vocabulary.

Word stemming is the process of truncating or modifying words in the hope of reducing complexity and finding the derivative forms of words with similar meanings²⁹. For example, the words “organize”, “organizes”, and “organizing” all share common meaning. This process removes plural and possessive suffixes from words and finds the root forms of words. In our example *response phrase* in Figure 17, this process transforms the phrase “roast-beef, pizza, hamburgers” to be “roast-beef, pizza, hamburger”. For our classifier, this breaks down possible responses into more generic, recognizable categories. Additionally, by reducing entities to their root forms our classifier is more likely to find images for these root forms, rather than the original complex or plural forms.

roast-beef pizza hamburgers

Figure 17: Response Stemming

Using these strategies, our classifier is able to recognize any n-gram word response, as well as any number of total responses in the phrase of text. We limited detected n-grams to reduce the computation required for a real-time application, and to simplify the presentation of responses to the user. Increasing the number of words when comparing n-grams increases the number of individual comparisons required on an exponential scale. Therefore to improve performance, we limited the classifier to only recognize tri-grams.

3.4 Image Classification and Retrieval

In order to present the most relevant image for each entity, our classifier queries a database of image labels that describe what is portrayed by each individual image. Livox maintains a database with this information, however, existing labels are not always accurate to what is in the image, and the labels do not account for compound-word entities. Additionally, Livox has many images that portray the same entity, with no way of finding the most relevant image for it.

We relabeled all of the images currently being used in the Livox application in order to present images for compound-word entities and to find the most relevant among many potential images. To achieve this, we used the Google Vision API, which applied multiple categorical labels to each image and produced confidence levels for each label. For example, the image of a dog in Figure 18a was classified with labels including “dog, mammal, vertebrate, and puppy” as well as confidence levels between zero and one.



(a) Example Image

Label	Confidence (%)
Dog	0.9953
Mammal	0.9890
Vertebrate	0.9851
Dog breed	0.9838
White	0.9692
Bichon frise	0.9566
Puppy	0.8547

(b) Google Vision API Generated Labels

Figure 18: Google Vision API Generated Labels For an Image

We entered these categorical labels, as well as a unique identifier for each image into an independent database. Our classifier queries this database using each response, and finds the image with the highest confidence in the label its querying for. For example, if it were searching for an image of a “puppy” entity, it would not likely use the image in Figure 18a since it’s only 85% confident that the image is indeed a puppy. However, if it were searching for an image of a “bichon” entity, we would be very likely to select the image in Figure 18a since we’re 95% confident that the image contains a bichon. The classifier then fetches the actual image from Livox’s existing database to be presented to the user.

4 Implementation Strategy

This section focuses on the tools and general strategies our team used for research, project management, and programming. We used Google Drive for organization and collaboration. The main tools we used to assist with communication and task management were Slack, WhatsApp and Jira. For programming, we utilized integrated development environments such as Android Studio and Pycharm, and organized code through GitHub and BitBucket. For rapid prototyping and testing of our list classification module, our team deployed a microservice architecture. This module interacted with numerous external APIs and services such as Google Cloud and Amazon Web Services (AWS).

4.1 Agile Development

We followed an Agile Development methodology to maximize our productivity while completing this project⁴³. Agile development emphasizes an iterative approach to software engineering focusing on several core values including:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

With these core values in mind, Agile developers conduct a cycle of iterative improvements to a piece of software, each ending with a working version of code. These cycles, often called sprints, typically last from one to four weeks depending on team size and the complexity of the project. Our team utilized a weekly sprint schedule, with each sprint starting and ending on Wednesday. Each sprint began with a team planning discussion, where tasks to be completed in the next week are formalized, prioritized, and given a subjective complexity score based on the difficulty of the task.

During the sprint, tasks are organized on a kanban board which limits the number of tasks that can be in progress at any time. This encourages developers to complete, test, and document a single task before starting a new one. Additionally, by using specific staging areas for test validation and code reviews, developers can ensure their code meets the highest standards of the team before being included in a deployment to the customer. In order to ensure tasks are completed smoothly, we conducted a scrum meeting each day, lasting no longer than 20 minutes. In each meeting we reported our progress on tasks within the previous day as well as any roadblocks we ran into. We then discussed the priorities of the team and set personal goals for work to be completed the next day.

At the end of the sprint cycle, developers produced a working version of their code, presented their changelog, and gave a demonstration to project stakeholders. This sprint review served as a chance for stakeholders to give feedback on the direction of the project and request specific pieces of functionality within the software. This unique approach allowed developers and stakeholders to cocreate a piece of software across many iterations, and through tight collaboration. Our teams, after review with stakeholders, conducted a sprint retribution to identify how the team can work more effectively and to increase the volume and quality of work in future sprints.

During sprint retribution, team members sum the complexity scores of all tasks completed within each sprint to identify a team velocity, used to estimate the amount of work done by the team as a whole. Team velocities can be compared between sprints to evaluate how the productivity of the team changes over time. An ideal team would slightly increase their velocity each sprint and become more efficient working together. However, this is extremely rare, and most teams aim to achieve a consistent yet challenging velocity across all sprints. Finally, after gathering feedback from stakeholders and discussing how the team can act more efficiently, teams begin the process of planning for the next sprint cycle.

4.2 Technology Stack

4.2.1 File Organization and Task Management

Google Drive is a file synchronization and document authoring tool⁴⁴. We used it to collaborate on research organization, meeting notes, and code documentation. It allowed us to all have shared and concurrent access to documents in order to collaborate in real time. Google Drive is free to use for a limited amount of storage, which is all we needed for this project.

Jira is an issue tracking and Agile product management tool used for creating, tracking and assigning tasks to group members⁴⁵. Jira was designed for Agile development so the organization of epics and stories is built into the framework of the task tracking software. Livox currently uses Jira for their own task tracking, however, we created our own backlog of tasks as to track our productivity independently from the rest of the Livox developers.

4.2.2 Communication

Slack is a convenient, real-time messaging platform designed for communication within project teams⁴⁶. We use Slack to communicate, share schedules, and organize daily activities within our WPI student team. As students, we are all involved in multiple group projects, and Slack provides functionality to switch between these groups in a single application.

WhatsApp is a messaging and voice over IP service⁴⁷ made for both mobile and desktop environments. We used WhatsApp as a high-priority contact method with our project sponsor Carlos Pereira and members of his engineering team such as André Camara. Many members of the Livox team are located in Brazil and WhatsApp is a popular solution for convenient and free international communication.

4.2.3 Programming

GitHub is a software development version control hosting platform⁴⁸. All of our code was created and shared through a private repository on GitHub, allowing us to maintain a stable, production version of code, while having additional branches to work on new features. We maintained a rule that no code should be pushed straight to production in order to reduce code merging errors and to maintain the integrity of our codebase.

Android Studio is the official Integrated Development Environment (IDE) for Android⁴⁹. Android Studio was used in developing our Java code and user interface (UI). We chose Android Studio because it offers useful Android development tools, such as high-level UI design, Git integration, dependency management, and instant deployment to virtual or USB connected devices.

Pycharm is an IDE for Python programming⁵⁰. We used Python alongside Java to expedite the time to completion for single-time tasks and to create quick, proof-of-concept applications. Our image labeling was implemented using Python scripts that connected with Google Cloud and our AWS database. We developed our proof-of-concept application and prototype list classification package using a Python web API to develop and test solutions quickly in a language with more support for AI and NLP tasks than Java.

Flask is a lightweight Web Server Gateway Interface (WSGI) web application framework for python developers⁵¹. This was used so that functionality could initially be written in Python before being implemented into Java, and data could be sent and received by the application while the code was not local to the tablet. Our Flask API was used for almost all functionality while the program was still online. After the application was taken offline, it was used for running tests to evaluate the classifier and to prototype new solutions.

4.2.4 External APIs

Amazon Web Services (AWS) provides on-demand cloud computing, database management, and file storage, along with many other services⁵². AWS offers services that allow for quick and inexpensive deployment of MySQL databases. We used AWS to host and manage a database of image labels created using Google Cloud’s Vision API.

Google Cloud provides a suite of cloud computing services with one of the most versatile image labeling APIs⁵³. We used Google Cloud to interact with the existing Livox image database and the Google Vision API. With a new account, we received free credit to use towards cloud services. Using this credit we labeled all images in the Livox database at no cost.

Microsoft Azure is a cloud computing service similar to both AWS and Google Cloud⁵⁴. We utilized Azure to explore the reliability of real time speaker identification for persistent interlocutor profiles. Azure offers models that can be trained to identify up to 10 distinct voices within a user profile.

Amazon Mechanical Turk⁵⁵ is a crowdsourcing marketplace that enables the outsourcing of simple tasks to a distributed virtual workforce. We utilized Mechanical Turk to generate a dataset of *list questions* through a distributed Google Forms questionnaire. Participants gave examples of *list questions* based on their own understanding of the classifier. We later used this dataset to measure the performance of our classifier and to identify areas of improvement for the future.

4.3 Microservice Architecture

Microservice architecture is a style that splits a singular application into a set of services, with minimal dependencies, that allow the program to be more modular⁵⁶. We divided the architecture of our list classifier into separate modules that have specific purposes: converting the speech into text, recognizing there’s a question, splitting the *question* and *response phrases*, and isolating the named entities. This implementation style allowed us to create minimum viable products for each section and then iteratively test and improve each module as needed. We aimed to start with simple solutions and expand complexity throughout our development. The list classification package as a whole also acts as a microservice, minimizing issues while integrating with Livox’s existing codebase.

5 Evaluation

The primary goal of this project was to extend the existing machine learning algorithms, within the Livox application, to further reduce the time and effort required to communicate for people with disabilities. We aimed to create a solution that could encapsulate phraseology used in natural conversation, and have strong accuracy for commonly asked questions.

Livox is a mobile application, and supports users both at home or school and while traveling. Therefore, it was important to verify that our classifier functions in offline environments without an internet connection, while users are away from their home or school. Additionally, Livox is a low-cost alternative to many other AAC solutions, therefore our classifier should efficiently run on low-end hardware to support older and more budget-oriented devices.

We evaluated our solution using multiple methods, each targeted at measuring the effectiveness of different aspects of our classifier. We tested our classifier to gain understanding for three main purposes, each with one or more methods of evaluation:

1. Data-Driven Development Decisions: To understand how our classifier can be improved and to align with the functionality described by Livox stakeholders
 - Targeted Testing of online and offline solutions
2. Quantitative Metrics: To measure the accuracy of our classifier's design
 - Crowdsourced Dataset Testing
3. Qualitative Metrics: To measure the real-world impact our classifier has for users of Livox
 - Image Matching Rating
 - User Studies

In this section we further describe the motivations and methodologies for each aspect of our evaluation.

5.1 Targeted Testing

We created a dataset of 30 targeted test questions, through collaboration with project sponsors, to ensure specific functionalities were achievable by our classifier. We used this set of questions to identify specific milestones in our classifiers ability to handle varying:

- Types of questions: Can ignore *yes or no* questions and all *non-list questions*, and recognize all *list questions*
- Sentence structures: Can parse sentences with varying phraseology and syntax
- Types of responses: Can extract compound-word and plural responses

We created a dedicated testing endpoint in our online API in order to evaluate the classifier against this entire dataset of questions. The test endpoint executed each of the API services across the collection of test questions and produced both detailed results for each question, as well as a summary for all cases. Our summary results include the number of questions that were successfully parsed, failed to be parsed, or caused an error in our processing. The detailed results include the output from each endpoint, indicated in Figure 19 below.

We conducted tests after the completion of four stages of development including:

Classify input text as a list question: <i>Question Classification</i>
What would you like for dinner; pizza or pasta? → List Question Do you want dinner? → Yes/No Question
Separate the question from the list of responses: <i>Phrase Parsing</i>
What would you like for dinner; pizza or pasta? → What would you like for <u>dinner</u> <u>pizza or pasta</u>
Extract the list of entities: <i>Entity Parsing</i>
 pizza or pasta → [pizza, pasta]
Retrieve relevant images for each entity: <i>Image Retrieval</i>
[pizza, pasta] → images

Figure 19: Modular Model Evaluation

1. Naive phrase parsing and entity extraction
2. Dynamic phrase parsing and entity extraction through word embedding
3. Improved phrase parsing and word stemming
4. Extended word embeddings and vocabulary

Many of these questions were created to identify weaknesses in our implementation. Some difficult cases were created, that stakeholders identified to be non critical, as stretch goals for our project. We additionally included cases for false positives, questions that our classifier would not recognize and parse, to identify how our classifier reacted to and recover from errors in processing.

5.2 Crowdsourced Testing

In order to gain an objective understanding of our classifier’s performance, we developed an unbiased dataset of *list questions* that contained minimal influence from our own understanding of how the classifier works. This data and subsequent analysis was used to measure how well we achieved each of our functional requirements. We created multiple survey questionnaires which were posted through Amazon’s Mechanical Turk (Appendix H). The goal of our survey design was to have a variety of initial *question words*, n-gram frequencies, conversation topics, and sentence structures. Each entry in the survey had three components:

1. The full question as it was written by the respondent. This excluded capitalization, punctuation, abbreviations that are not spoken as such, and numeric or other special symbols.
2. The possible responses as written by the respondent. The tiles that should ideally appear on screen for selection.
3. The conversation topic. These can be categorized as *activity*, *date/time*, *description*, *locations*, *object/entity*, *organization*, *number*, *people*, or *miscellaneous*.

We created multiple batches of surveys, each including a brief description of how to formulate *list questions*, along with two or three examples depending on the survey batch. We included minimal information and simple examples in the description to acquire questions from individuals with minimal knowledge of how the classifier works.

In our first two batches, we found that respondents used the same question initiators as in the examples provided. For example, our first round used the *question words* “what”, “do”, and “how” while the second used “are”, “should”, and “is.” As seen in Figure 20a and Figure 20b, the survey responses were skewed towards the examples used in the description. We identified one exception to this rule, the *question word* “do,” which is used heavily in each case despite not appearing in any example in the second batch.

We also found that respondents used the same conversational topics as in the examples provided. For example, our first batch of questions only asked about *objects/entities*, while the second asked about *objects/entities*, *people*, and *organizations*. As seen in Figure 20c and Figure 20d, the distribution of topics is biased towards these examples, and that varying the topics used in examples did produce a more balanced distribution between categories.

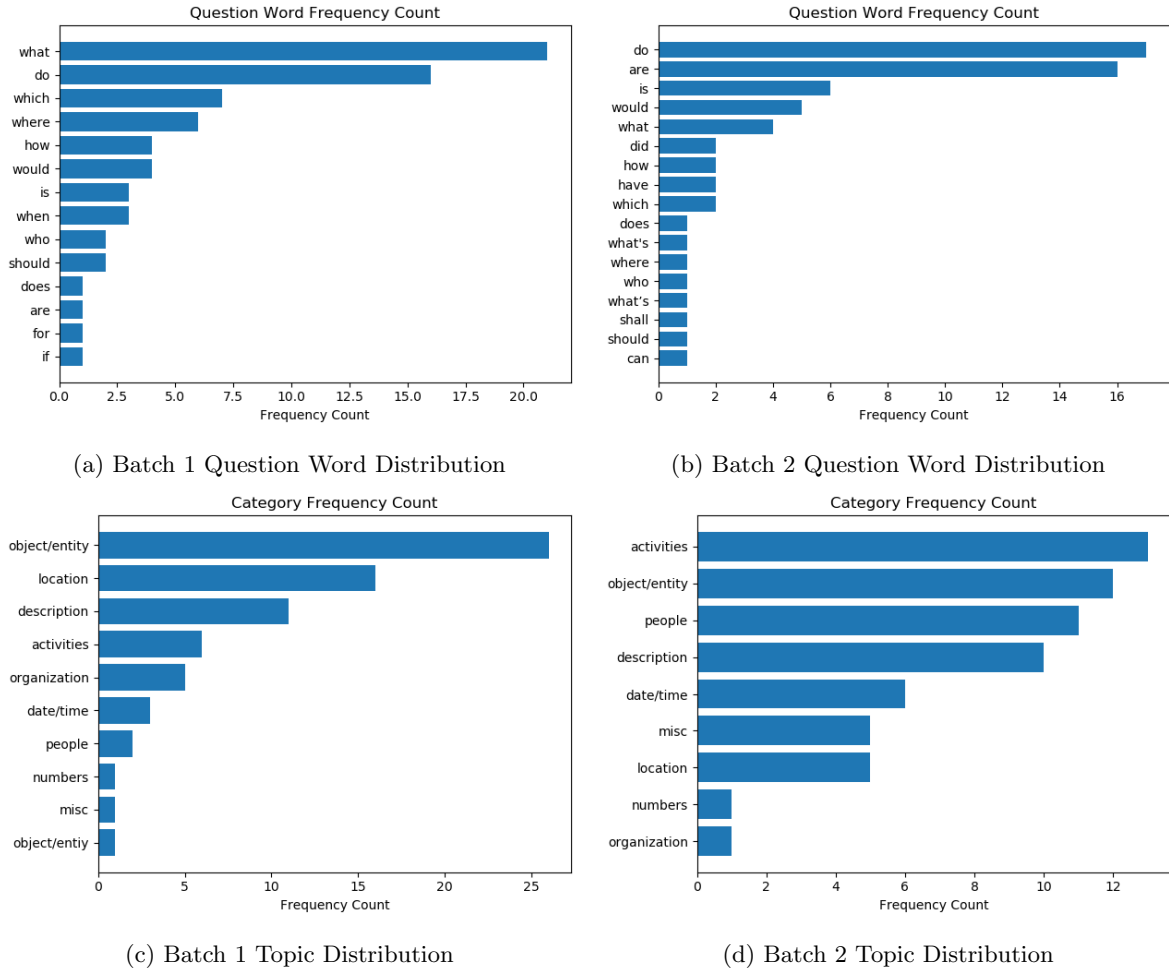
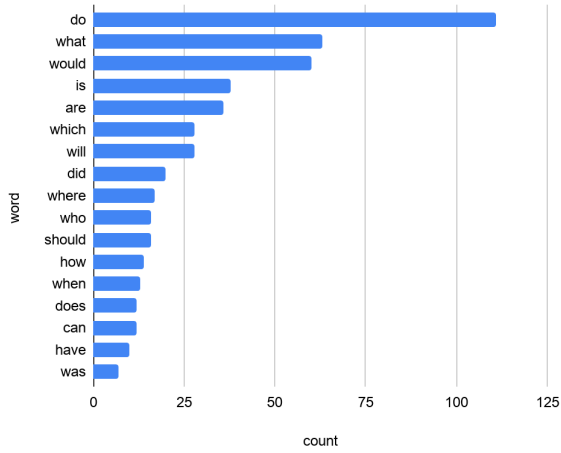


Figure 20: Mechanical Turk Batch Results

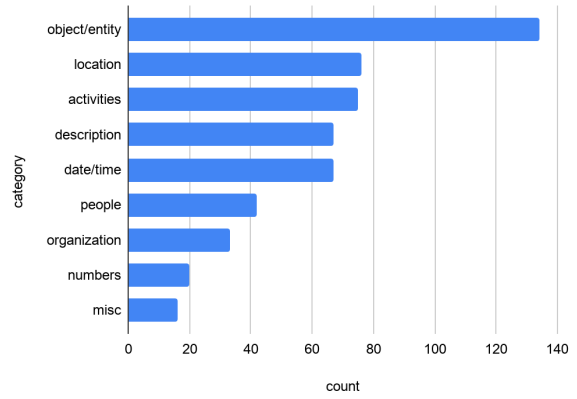
Based on these findings, we varied the examples used in later batches as to gain a better distribution for both *question words* and conversation topics. We also encouraged respondents to use a variety of conversation topics, a variety of n-grams, and a variety of number of response options. After making these changes, we conducted an additional three batches of questionnaires, each with varied examples.

We collected a total of 530 *list questions*, each labeled with the expected responses, and the topic of conversation. The overall distribution of *question words* in Figure 21a shows a balanced frequency of words used, except for “do”, which is used far more frequently than others. The distribution of conversation topics in 21b shows a similar distribution to Batch 2, where *objects/entities* and *activities* are among the most common topics, *description* and *date/time* is mid-range, and *numbers* and *organizations* are among the least common. Similarly, the distribution of n-grams in Figure 21c shows a good variety of compound word responses provided.

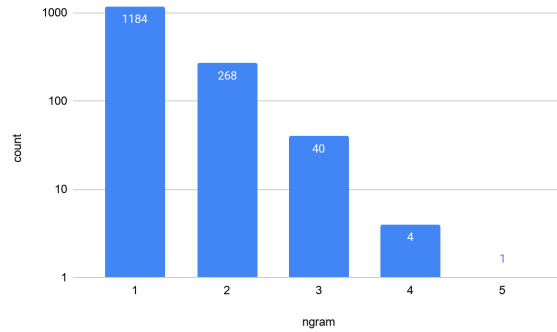
Once the dataset was collected we conducted cleaning, editing and modifying the questions to be in a format recognizable by our online API endpoint. Our cleaning modifications were limited to standardizing capitalization, removing incorrect punctuation marks, and correcting the spelling of misspelt words.



(a) Final Question Word Distribution



(b) Final Topic Distribution



(c) Final n-gram Distribution

Figure 21: User Study Results

We combined this dataset with a collection of *yes or no* questions from a BoolQ dataset⁵⁷ collected in 2019. We included 500 samples from this dataset in order to identify how well our classifier could recognize and ignore questions outside its capability. This augmented our evaluation dataset to include a total of 1030 questions of varying types.

Once the dataset was cleaned and augmented, we ran the set of questions through our online API test endpoint. We collected several metrics from the results of our test, including

- Overall Precision: Fraction of questions that are correctly parsed, and for which relevant images are found
- Balanced Accuracy for Conversation Topics: Averaged fraction of successful tests for each class of conversation topic

5.3 Image Matching Evaluation

In order to evaluate the relevance of images presented to users, we rated 137 pairings of responses and images. These responses were extracted from the expected entities specified by Mechanical Turk respondents, rather than using the entity parser endpoint of our API. This ensured the entities matched the respondents actual intention, in the case that our API did not properly parse the full question.

Entities were extracted from the nine possible question categories as reported by survey respondents: *object/entity*, *activities*, *numbers*, *description*, *location*, *date/time*, *organization*,

miscellaneous, and *people*. At least 10 entities of each category were selected to ensure meaningful and significant conclusions were made.

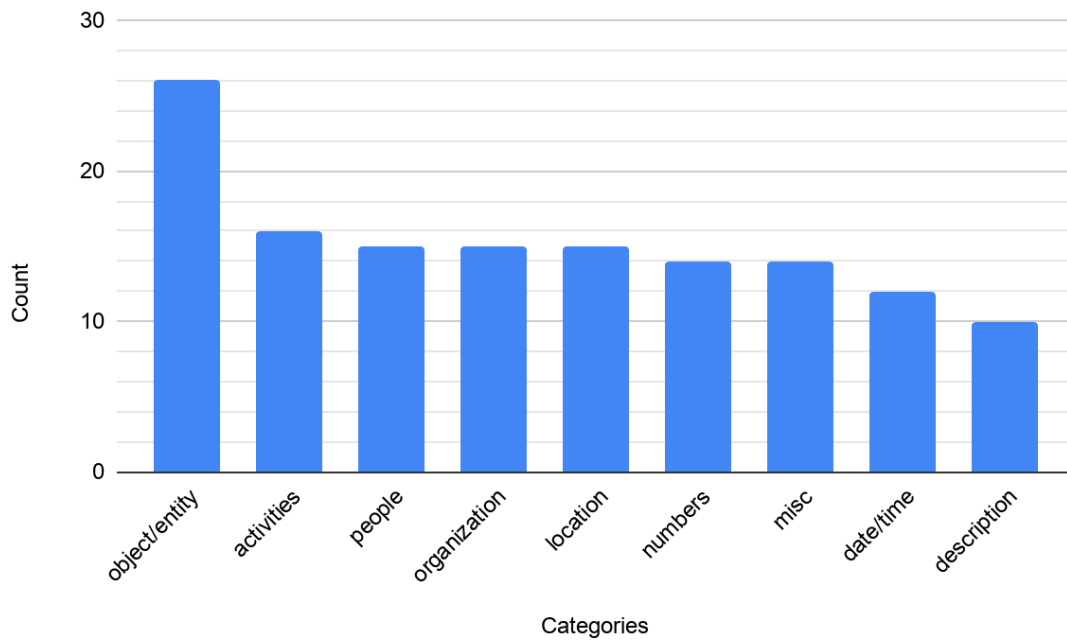


Figure 22: Image Rating Topic Distribution

We individually rated an image based on how well it represented the related response on a scale of one to seven (1-7).

- (1-3) Ratings indicate images that do not effectively represent the response or are irrelevant to the conversational scenario
- (4) A rating of four was reserved for the case that no image is found.
- (5-7) Ratings of five to seven indicate images that effectively represent the response in the given conversational scenario

Our API failed to find images for some responses, resulting in the default image of a question mark being returned. Not all entities or responses may have an image within the Livox application which can describe them. Therefore, it is preferable that the feature present no image rather than an inaccurate one in these scenarios.

Examples of each rating can be seen in Figure 23 below. We then calculated the mean opinion score (MOS) across all pairings of entities and images to gain an overall understanding of our image relevance.

5.4 User Study

We conducted two studies to evaluate the usability and effectiveness of the developed feature for users of Livox. These studies gathered data to describe the overall reduction in the reciprocity gap between users and interlocutors. Therefore we collected data for each involved individual in the conversation, Livox users and interlocutors (Appendix G).

The *user set* was made up of two stages to test time and effort to communicate, each with our classifier disabled or enabled, in order to measure the improvements it provides. The *interlocutor set*








Score of 1	Score of 2	Score of 3	Score of 4	Score of 5	Score of 6	Score of 7
Three (<i>number</i>)	Fall (<i>date/time</i>)	Breakfast (<i>activities</i>)	Minecraft (<i>misc</i>)	December (<i>date/time</i>)	Retriever (<i>object/entity</i>)	Tacos (<i>object/entity</i>)
						

Figure 23: Average Scored Examples

used three stages to test interlocutor usability, providing participants with increasing levels of knowledge at each phase.

We recruited 20 individuals both on and off campus using a snowball sampling method. We gathered initial participants through the use of flyers and social media communication, and used this initial participant pool to connect us with peers who would be interested in participating in our study.

We did not recruit any participants with disabilities or impairments, although this is the primary demographic of individuals that use Livox. We did not recruit any individuals with disabilities because the test can still accurately evaluate the number of physical interactions and comparative time taken to select the desired option. Both these metrics contribute to the reciprocity gap faced by Livox users and interlocutors, therefore, the feature is successful if they are both decreased from the current method of navigating the application^{neausi}.

5.4.1 Interlocutor Set

After explaining the purpose, functionality, and intended usability of Livox, the subject was instructed to ask the application a series of questions for our classifier to analyze. The first set of questions was exploratory, allowing the participant to interact with the classifier, having minimal knowledge of the sentence structures it expects. The goal of this testing was to allow the user to explore the tool without influencing them into specific types of interactions⁵⁸. In this phase, we recorded whether the question activated our list classifier, and the number of attempts it took to do so. We additionally recorded whether the question activated the *yes or no* classifier already built into Livox when this kind of question was asked.

The goal of the second stage was to evaluate the accuracy of our classifier for known *list questions*, but with varying voices and inflections from different participants. To do this, participants were given a list of scripted questions which were read aloud. Similarly to the first phase questions, we recorded whether the question successfully activated our classifier in the Livox application, and the number of attempts necessary to achieve this.

The third phase was similar to the first phase, where the subject came up with three new questions to ask. At this point in the study, we explained the sentence structure the classifier expects, and answered any questions the participant asked about how to properly formulate *list questions*. We again recorded whether the question activated our list classifier, and the number of attempts it took to do so. We aimed to achieve similar results to our second phase from our third phase, because participants now had a strong understanding of our classifier and how it functions.

5.4.2 User Set

For the two parts of the *user set* we asked the subject a series of predefined questions, and the subject responded through the Livox interface as if they were impaired and could not communicate

otherwise. For the first phase, our voice-activated classifier was disabled, and the user had to select their answer by navigating the convoluted Livox nested menus. In order to estimate the performance, we recorded the time between asking a question and the question being answered, as well as the number of button presses required by the user. For the second step, our classifier was enabled, and the user responded by selecting the desired response from those returned by our classifier. The same metrics were recorded. To evaluate the performance of this set, we compared the time and number of button presses required to answer questions between the two phases.

6 Iterative Development

This section focuses on the technical evolution of our project, as opposed to the project methodology, which covered the created solution more generally. Early iterations focused on the development of an online API hosted on AWS for our NLP microservices. Concurrently, we created a standalone Android application to connect to the API and connect with Livox’s current speech recognition package. We later integrated this functionality with the existing Livox application and ported our NLP solutions to work offline on the Android tablet. We created five microservice modules to achieve this solution:

1. Question Classifier
2. Phrase Parser
3. Entity Parser
4. Image Classification and Retrieval
5. Android Application

In this section, we discuss the initial solution for each of these microservices, describe the iterative improvements we made, and provide evidence for why we made the associated design decisions. We also discuss the design and development of a standalone, prototyping Android application, as well as a fully-integrated beta version of Livox.

6.1 Question Classifier

The question classifier recognizes if a recorded piece of speech is a *list question*. We developed three different solutions to achieve this, however, we only used the last solution developed, as it is the most accurate and aligns best with our functional requirements.

6.1.1 Iteration 1 - Hard-Coded Phrase Recognition

In the initial implementation of the prototyping application, the question classifier was limited to recognizing hard-coded phrases made up of a question initiator phrases and prepositional phrases, such as “what would you like” for the question initiator and “for lunch” as the prepositional phrase. These components act as a combined *question phrase*, and our classifier only parsed text including these exact *question phrases*. This solution obviously did not reach our functional requirement of encapsulating natural conversation. The phrasing was too restricted to allow for intuitive interaction with the developed feature. However, we used this solution to test and better understand the flow of data through the Android Speech Recognition package and to familiarize ourselves with Livox’s existing speech-to-text functionalities.

6.1.2 Iteration 2 - SVM Question Classification Model

Our next iteration focused on creating a generalized solution using machine learning. This solution utilized a support vector machine (SVM) to classify questions. We prototyped this solution initially using a microservice architecture through our Python web API, in order to offload the computation to cloud computing services. We used TF-IDF vectors to represent each question, and to transform questions into a format recognizable by our classifier. Our SVM then used the frequency vectors generated for a question in order to identify the class of the question. We trained our model using a UCSD question and answer dataset⁵⁹, providing around 1.4 million question and answer pairs scraped from Amazon product reviews. The dataset labeled each question as either a *yes or no* or an open-ended question. We trained our model to accept all “open-ended” questions and to ignore all *yes*

or no questions since Livox already has tools implemented to recognize these questions. We trained our model using a sample of 4,000 questions from the 80,496 questions relating to “Health and Personal Care” products, and found that increasing the number of training questions beyond this did not improve the model’s accuracy or generalizability. This is the same dataset Livox used to train their existing *yes or no* classifier, so we believed it would give consistent performance for our application. This solution achieved a precision of $p=0.75$ on our set of targeted testing questions. Although this was relatively good performance, the classifier mislabeled many of the key examples we considered necessary. An additional limitation of this solution was its computational complexity, meaning it may not perform efficiently on low-end devices. The heuristic-based solution of our next iteration solved both of these limitations.

6.1.3 Iteration 3 - Heuristic Method

In order to validate the accuracy of our machine learning model, we created a heuristic-based model to classify text as a *list question*. This model searches for the word “or” and a *question word*, such as “who”, “what”, or “why”. We maintain a list of *question words* that can be loaded as the application starts. This allowed our team to add examples as we continued our targeted testing. The two components searched for indicate the presence of a response and *question phrase*, respectively. Our heuristic model identified every *list question* we generated for our targeted testing, significantly outperforming our SVM based solution. The increased accuracy and reduced computation led us to use this solution rather than the SVM model.

6.2 Phrase Parser

Once a question is identified as a *list question*, the *question phrase* must be separated from the list of responses. We initially explored a machine learning solution to achieve this based off of the Amazon product review dataset of questions. However, this solution performed very poorly and there was no other pre-existing text dataset that could be used to train such a model. Additionally, creating our own dataset would have been too time-consuming for the duration of this project. Therefore, we used a heuristic-based approach, and improved it as needed to achieve our functional requirements.

6.2.1 Iteration 1 - Common Question Words Offset

The first iteration of the phrase parser relied on two steps of processing. The first phase used a list of *question words* to distinguish the beginning of the *question phrase*. We started with five common *question words*, “who”, “what”, “when”, “where”, and “how”. The second phase of parsing used a list of prepositions and a numerical offset to distinguish the end of the *question phrase*. We used the numerical offset to predict the number of words until the prepositional phrase will end after the preposition itself. Together, this method could find the beginning and end of the *question phrase*, and separate the question from the list or responses. This solution handled common phrases well, but lacked generalizability for natural language, as it did not include many question initiators for the first phase of parsing. Additionally, we occasionally predicted too high of a numerical offset, removing important information from the *response phrase*, ultimately resulting in too few or incorrect entities being extracted.

6.2.2 Iteration 2 - Improved Question Words

To address the issue of poor generalization in the first phase of parsing, we added more *question words* to the list being used, such as “do”, “is”, and “are”, totaling to thirteen *question words*. To address the issues of entities not being included in the list of responses, the word offsets were changed to estimate the end of the *question phrase* more conservatively. By including more of the text in the *response phrase* than the question, we choose to rather recognize more rather than fewer entities. Finally, in cases from our targeted testing without recognized questions words, we used a default offset

value of four words from the start of the sentence. These changes led us closer to our functional requirements of accurately representing each of the responses in the question, and encapsulating natural conversation.

6.3 Entity Parser

Once the *question* and *response phrases* are separated, the entity parser is responsible for extracting all possible responses to the question from the list. We developed five iterations to achieve our final solution, each improving our classifier’s accuracy and performance on our targeted test questions.

6.3.1 Iteration 1 - Naive Entity Extraction

Our first iteration removed all stop-words from the *response phrase* using NLTK for Python. It then extracted an entity for each individual word in the list. The limitation of this solution was that it did not recognize compound words as a single entity. For example, in Figure 8, it would not recognize “roast beef” as a single entity, rather, it would recognize the two entities “roast” and “beef.”

6.3.2 Iteration 2 - Word Embedding Vector Similarity

In order to recognize compound-word entities in the *response phrase*, we used Word2Vec to calculate similarity scores between single words and combinations of words grouped together. We calculate the cosine-similarity between all single words, pairs, and triplets to find the most related combination of the words in the *response phrase*. For the example phrase in Figure 10 the combination of words “roast-beef, pizza, hamburger” makes much more sense, and is closer in the embedded vector space, than the combination of words “roast, beef, pizza-hamburger.” The combination with the highest similarity between entities is chosen as the final list of entities. This solution reliably identified the correct compound-word entities in our targeted testing. However, the limitation of this solution was that it violated our non-functional requirement to work on low-end devices. The word embedding model took up over four gigabytes of memory which is not reasonable on low-end devices, often having no more than 500 megabytes of memory, which was the case with our hardware used for development and testing.

6.3.3 Iteration 3 - Database Derived Vocabulary

The goal of our third iteration was to reduce the space required by the model in memory or in device storage, while maintaining a similar accuracy to that of the larger model. To achieve this we derived a vocabulary from labels generated by the Google Vision API on all the Livox images. Google Cloud’s Vision API allows for individual and batch labeling of images, as well as producing confidence values for each label. We used this API to label the entire dataset of Livox images and created a MySQL database structure to connect the newly defined labels to the existing Livox image database. These labels were used to create a vocabulary of recognizable compound-word entities in the database

This approach significantly reduced the number of responses recognizable responses, because it was limited by the number of images in the Livox database. However, this solution was still capable of recognizing most common entities and identified all common n-grams in our targeted testing dataset. The amount of space used in memory was reduced significantly, only requiring about 20 megabytes of space either on disk or in RAM. This solution achieved our non-functional requirements of running on low-end devices and in offline environments. However, the limitations of this solution were related to the reduced vocabulary compared to the larger word embedding model and lack of recognition for plural entities, such as “cars” or “french fries”.

6.3.4 Iteration 4 - Pre-processing with Word Stemming

Our fourth iteration focused on pre-processing the text data before trying to extract entities from it. Word stemming served as a lightweight, heuristic-based addition to reduce words to their root forms before extracting them. Using this technique, our parser could detect and remove plurals and suffixes of stem words. This iteration brought us closer to encapsulating more natural conversation, however, the limitation of this solution was that it still relied on the reduced vocabulary of our classifier and could not fully attain our functional requirements.

6.3.5 Iteration 5 - Word Embedding Extended Vocabulary

The final iteration extended the vocabulary to recognize words that were not included in the database tags, but that could be related to at least one image in the Livox database. In order to connect responses to images that were not labeled as such, we utilized word embedding to identify the ten most related synonyms for each entity in the existing vocabulary, from the larger four gigabyte model previously used in our second iteration. With this addition, our vocabulary supported a much richer variety of entities closer to Google’s own word embedding model, without the size limitations for low-end devices. This final solution detected n-grams more accurately and was able to recognize informal slang words, such as “jammies” more formally as “pajamas.” This solution met both our functional requirement of encapsulating natural language and non-functional requirement of working on low-end devices and in offline environments.

6.4 Image Retrieval

Livox maintains a database of over 26,000 images that are included in the Android application. These images have labels associated with them, but are not accurately descriptive and cannot be used to identify individual images. The original labels contained few descriptor words and only one word for each label, separating n-grams into individual words. Relying only on the existing Livox labels would have limited the vocabulary and made querying the database of images difficult.

Therefore, we created our own database of labels associated with each Livox database image using Google Vision. We initially hosted this database through Amazon AWS, which could be easily interfaced with our AWS online API. However, in order to meet our non-functional requirement to work in offline environments, this database was localized on the tablet. Google’s libraries for Android applications included an interface that could be used to initialize and interact with a database running natively on the tablet. Therefore, this was integrated without any modifications to the existing Livox codebase.

6.4.1 Iteration 1 - Google Vision tags Image Retrieval

The first iteration retrieved images simply by querying the response against the labels in our Google Vision generated database. Each label associated with an image has an associated confidence interval. The retriever simply selected the image with the highest confidence interval for labels matching the response word.

The limitation of this solution was that it did not always present the most relevant image for each response, as recognized during testing. For example, an image of a shower drain, seen in Figure 24, was labeled as “paws” by the Google Vision API, showing that it would occasionally wrongly identify images. In other cases, we would know that the Livox application has images to represent certain entities, however our labels alone could not encapsulate them. Therefore, in our next iteration, we created a solution that retrieved images based on the labels in both our own and in the original Livox image database.

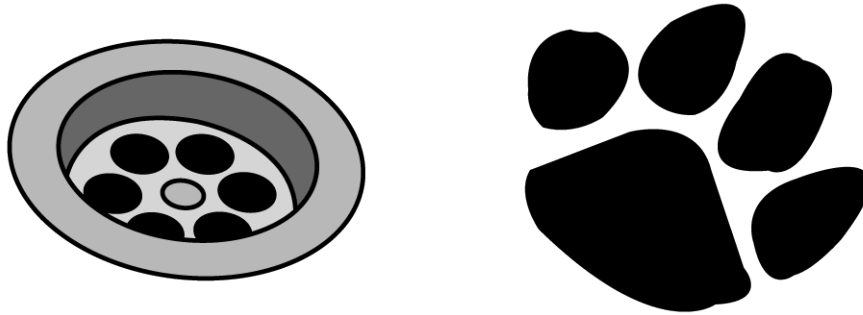


Figure 24: Returned drain labeled as paw by Google Vision vs Livox tagged paw

6.4.2 Iteration 2 - Score-Based Image Retrieval

In order to retrieve the most relevant image for each response, we developed a solution based on a relevance score for each possible image. This score is generated by querying both Livox's existing image labels, as well as the labels created through the Google Vision API. Each response is used to query the Google Visions labels, and when matched, is given a relevance equal to the confidence label provided. The relevance score is incremented further, by a constant factor, if Livox's original labels match the response. Finally, our classifier selects the image from the collection labeled to match response, with the greatest combined relevance between the two databases.

This solution worked very well for our targeted testing, and could be used to find images for all responses recognized by our entity parser, which relies on a dictionary of entities generated through the same Google Vision labels. This solution was also efficient enough to run on low-end devices and to run in offline environments as to meet our non-functional requirements.

6.5 Android Application

In the development of our project, we created a standalone Android application as well as a version of Livox integrated with our feature. These two applications were created respectively in the prototyping and integration phases of our project.

6.5.1 Prototype Application

Prior to this project, our team did not have any experience with Android development. Although we were all confident Java developers, we did not have any experience with Android application design and UIs. Therefore, we created a prototyping application to become familiar with these two key parts of our project.

The design of this application focused around creating a single Java package that could encapsulate all the code required for processing *list questions* and later be ported to Livox. We designed this package using the facade design pattern, creating a single interface class that delegated certain functionalities to other classes in the package. We used this application most heavily when testing for specific pieces of functionality outlined in our targeted test cases, and used the results to guide our design decisions for future weeks to improve the functionality of our API. The UI design focused on our ability to dynamically display images for questions that were asked and processed. Our initial UI was limited to displaying only two images for our hard-coded entity recognition. However, as we implemented additional functionality to our online API, we increased the number of possible images that could be displayed. Once we had created the intended Java package and UI, we began planning how we would integrate this into the existing Livox application.

6.5.2 Livox Integration

We designed our integrated natural conversation tool to act as independently from the rest of the Livox application as possible, following our microservice architecture goal. Our integration strategy only required a slight modification of two classes within the SpeechRecognition package and the main UI activity that coordinated launching other UI components and the multiple natural conversation tool's fragments. The list classification package was ported as-is to Livox, using the facade design pattern to act as a single interface for integration (Appendix A).

We designed the UI for our integrated feature to act as similarly as possible to the existing natural conversation tools being used in Livox. While the other tools focused on presenting static images for users to select, our tool required displaying images dynamically for users to select. Therefore, we modeled this dynamic behavior based on the implementation strategy used for the regular Livox screen, displaying an N by N grid of images for users to select. We designed our dynamic grid to be the same size and style as the user-defined Livox grid, to ensure as many of the Livox usability features can be taken advantage of by our tool. Although we were not able to integrate all usability features into our dynamic grid, for example, IntelliTouch and high-contrast images, we were able to integrate many and describe to more experienced Livox developers how these additional functionalities can be achieved.

7 Results

We collected results for each stage of our evaluation: targeted testing, automated testing, image relevance scoring, and our user studies. The results show that our classifier is accurate at classifying a variety of questions, and that it does reduce the time and effort to communicate for users of Livox. In the following section we report and discuss the results of our evaluation.

7.1 Target Test Results

Our targeted tests were conducted at the end of four iterations in order to measure progress in the development of this classifier. We added questions to our dataset partway through the development of our classifier to account for change in scope and to include tests more relevant to conversational scenarios suggested by project stakeholders. Tests in early iterations showed poor results, while tests in later iterations showed improvement.

Test	# Test Questions	Correct	Incorrect	Errors	% Correct	% Failed (Wrong/ Error)
1	27	1	26	0	4	96
2	27	9	10	8	33	67
3	30	18	10	2	60	40
4	30	21	7	2	70	30

Figure 25: Targeted Testing Results

Based on the results of our targeted testing, both developers and project stakeholders were satisfied with the functionality of our classifier. The classifier was able to detect and parse all *list questions* considered to be part of the core functionality of the classifier as described by stakeholders. The remaining two questions that reported errors are not *list questions*. Therefore, they were rejected by the classifier and are considered to be true negatives. The remaining incorrectly parsed questions contained phraseology intended to be stretch goals for our classifier. We hope that the remaining questions can be targeted and encapsulated by future work on this classifier.

7.2 Crowd-sourced Testing Results

Our crowdsourced test cases were additionally run through the online API test endpoint. The results of these 1,030 *list* and *yes or no questions* were used to measure how well we achieved each of our functional requirements and to identify how the classifier could be further improved.

The first part of our test endpoint ran each question through our question classifier service. This service labeled each question as either a *list question* or a *yes or no* question. The evaluation from this service, seen in Figure 26 show very positive results. The service successfully classified all 500 *yes or no* questions correctly, and classified 473 of the total 530 *list questions* correctly. Therefore, we report an overall accuracy of 94.46% for this service indicating that our classifier is very good at recognizing *list questions* and ignoring all *non-list questions*. This is extremely important as Livox already has a classifier to handle *yes or no* questions, and will implement additional classifiers to encapsulate other aspects of natural conversation.

The second step of our test endpoint ran each of the *list questions* through our phrase parsing and entity parsing services. The phrase parser split each question into a *question phrase* and a *response phrase*. The results of our phrase parsing were categorized into three types of splits:

- *Correct*: The *question phrase* and *response phrase* were correctly split.
- *Early*: The phrase parsing split in the *question phrase* before it ended. Extra words from the *response phrase* are then included in the *response phrase*. This is preferable to late splitting, as extra incorrect responses are not considered a major problem.

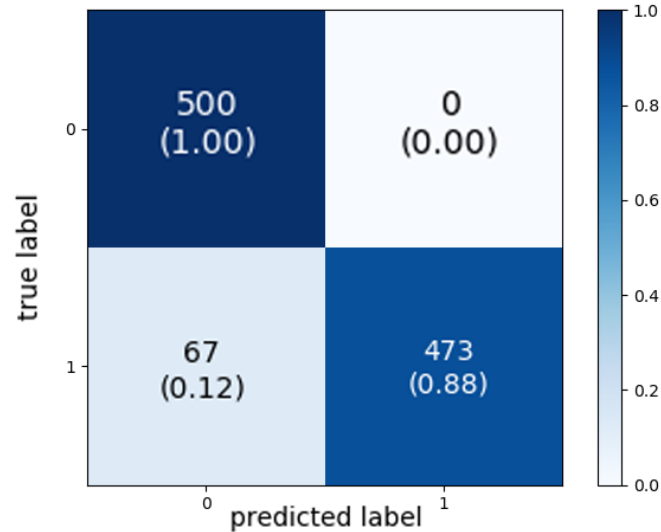


Figure 26: Question Classification Confusion Matrix

- *Late*: The *response phrase* was cut short, part of it was put into the *question phrase*. This is the worst outcome, as this could cause possible responses to be omitted.

The *response phrase* was then sent to the entity parser, where the possible responses to the question were extracted. These results were categorized into four groups based on the number of entities found. For each group, it is possible that extra entities not in the expected set were extracted.

- *All*: All of the expected entities were extracted.
- *Majority*: More than half of the expected entities were extracted.
- *Minority*: Less than half of the expected entities were extracted.
- *None*: None of the expected entities were extracted.

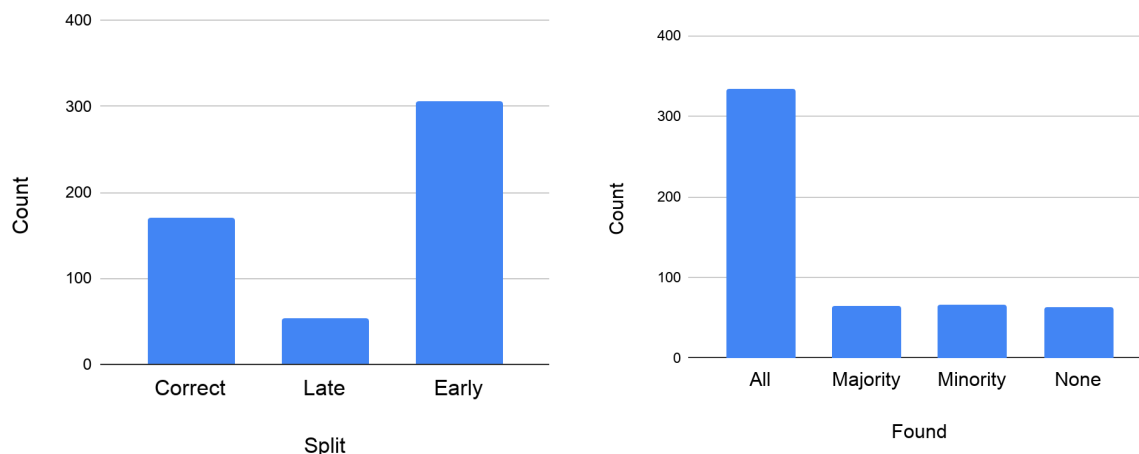
“What would you like for lunch pizza pasta or a hamburger”			
Phrase Parsing Categories		Entity Parsing Categories	
Correct Split	What would you like for lunch pizza pasta or a hamburger	Found All	{pizza, pasta, hamburger}
Early Split	What would you like for lunch pizza pasta or a hamburger	Found Majority	{pasta, hamburger}
Late Split	What would you like for lunch pizza pasta or a hamburger	Found Minority	{hamburger}
		Found None	{}

Figure 27: Phrase and Entity Parser Results Examples

The results of our entity parsing and phrase parsing services can be seen below in Figure 28a and Figure 28b We found only 32.3% of questions were split correctly, however, we designed our

classifier to split phrases early rather than late. This strategy improved usability as minimal relevant responses were missed. Our classifier removed relevant responses only 10% of the time, improving the results of our entity parsing service.

We found our entity parsing service extracted all correct entities 63.3% of the time, found all or the majority of entities 75.6% of the time. Our classifier extracted none of the responses only 12% of the time, showing our classifier can reliably extract both unigram and bi-gram responses from *list questions*.



(a) Phrase Parsing Results

(b) Entity Parsing Results



(c) Entity Parsing Accuracy by Category

Figure 28: Phrase and Entity Parsing

We further analyzed the entity parsing results based on the conversation category of the question in order to identify topics for which we have particularly strong or weak accuracy. When analyzing Figure 28c, the distribution of fully correct responses is balanced between categories, with the exception of *people* and *activities* being the lowest. This is likely due to the fact that there are very few, if any, images and labels associated with these categories. The user-defined correct responses for *activities* were often verb phrases, such as “play videogames”, rather than nouns, such as “Golden Retriever”. Finally, the *people* category often contained full names, such as “John Doe”. These names should be captured as a single entity similar to bi-grams, however our classifier does not have a list of common full names as it does with common bi-grams.

7.3 Image Matching Results

The goal of our image matching evaluation was to measure the relevance of images presented to users for different conversation categories. Each researcher rated the relevancy of an image to the response on a scale from one to seven for a total of 137 pairings. The results from this are positive, with an overall mean opinion score (MOS) of 5.20, and 68.1% of responses having a relevant image. In addition, no image was found for a given response 13.3% of the time, and an irrelevant image was found only 18.6% of the time.

For most response categories, the rate of irrelevant images was even lower at roughly 10% to 15%. However, *locations*, *date/time*, and *numbers* had a much higher rate of irrelevant images at 25%, 39.6%, and 51.8%, respectively. However, *locations* and *date/time* still had positive mean opinion scores, 4.70 and 4.20, respectively. Only *numbers* had an overall negative MOS of 3.61.

The categories of *organization* and *miscellaneous* had a high rate of not finding an associated image. This occurred 33.3% of the time for *organization*, and 44.6% of the time for *miscellaneous* entities. These categories had MOS's of 5.28 and 4.84. Not finding an image is not necessarily a problem, as the application has a limited number of images that can be associated with any response. Given these categories, it is expected that only the most common organizations have an associated image, so the failure to find images should be more frequent than for other categories. *Miscellaneous* entities included slang phrases, phrases from popular culture, or parts of speech that did not fit any other category. It is therefore expected that the image database would not recognize or contain images for these responses. There may be cases where no image is found, while a relevant image does exist. However, our image matching process avoids incorrect images by preferring false negatives, finding no image, over false positives, finding an irrelevant image.

The categories with the highest success rate were *object/entity*, *activities*, and *people*. These categories had mean opinion scores of 6.14, 5.92, and 5.78, respectively. In addition, they were associated with relevant images 90.2%, 82.8%, and 80.0% of the time. These categories are similar to the main tiles often featured on Livox's default home page, indicating that they are important in day-to-day speech. However, *date/time* is another category featured heavily in Livox, but did not have a definite positive mean opinion score.

Overall our image retrieval relevance shows mixed results. It is clear this aspect of our classifier can be greatly improved in the future by allowing users to suggest modifications to the image label database, although it is effective for common words in conversation.

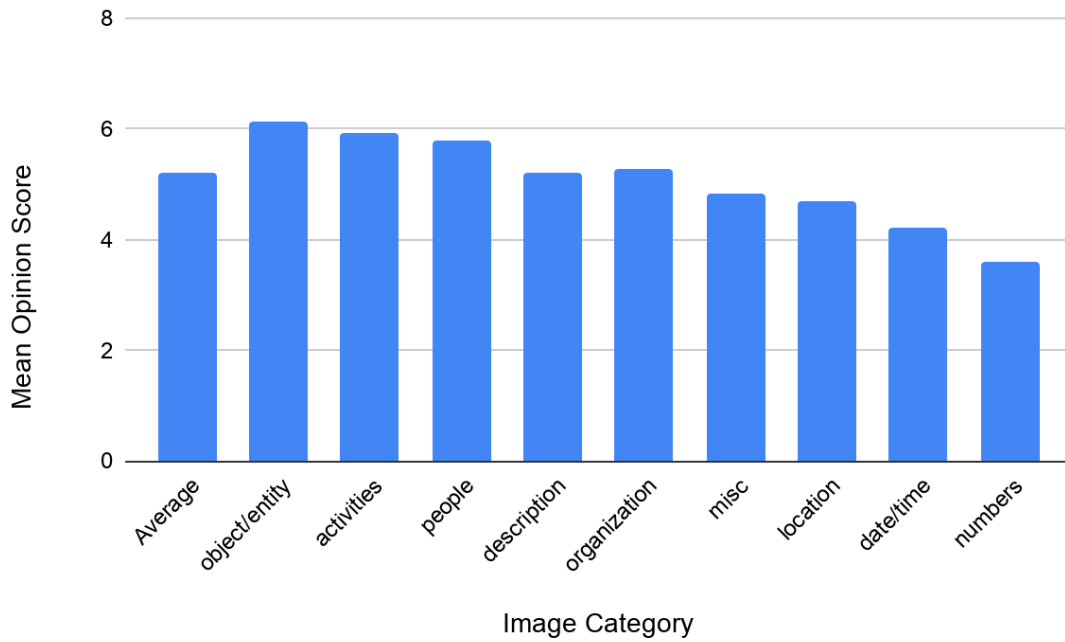
7.4 User Study Results

The user study showed that our feature was intuitive to use for interlocutors and effective at reducing the reciprocity gap for users.

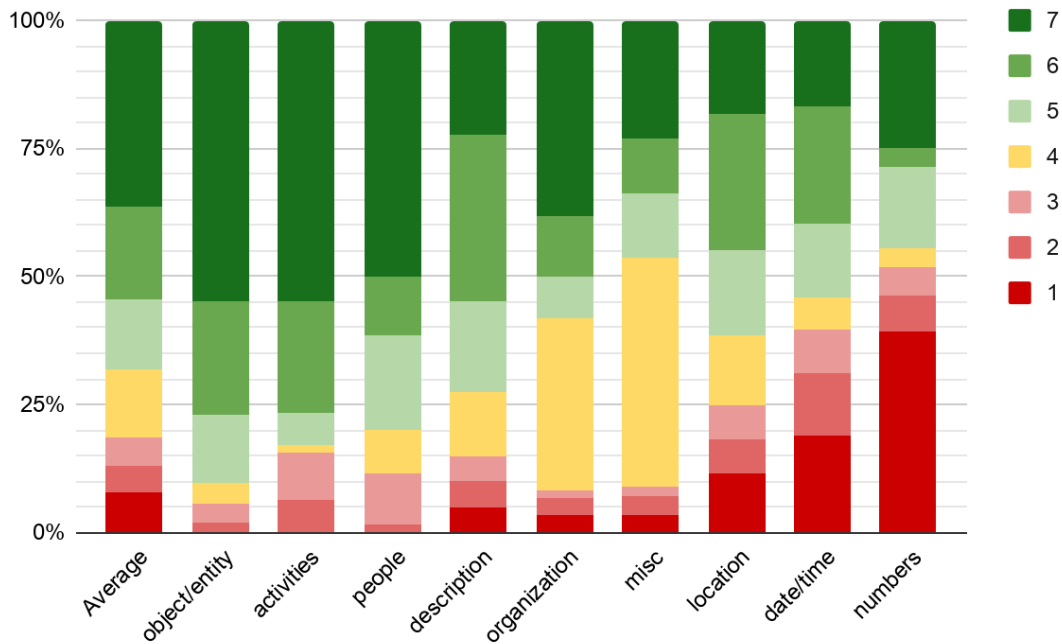
7.4.1 Interlocutor Interview Results

The results from the *interlocutor set* of our user study show that our classifier can reliably recognize known *list questions*. Additionally, the results show that participants can quickly learn how to formulate their own questions to be recognized by the classifier.

The uninformed exploratory phase tested whether participants, with no knowledge of the classifier, could formulate a question within its scope. In this phase, we simply had participants formulate and ask questions based on a set of conversation topics. Participants were only told that they were required to begin their question with the wakeword and to speak clearly without pausing. The results from this phase indicate that participant's questions could not reliably activate the classifier with minimal knowledge. The overwhelming majority of questions asked in this phase caused no actions from the list classifier within the Livox application. Some of the participant's questions inadvertently activated the *yes or no* classifier.



(a) Mean Opinion Score for Response Categories



(b) MOS Ratings by Category

Figure 29: Image Relevance Results

The scripted phase tested whether the classifier would reliably activate for known *list questions* when asked with a variety of voices and inflections. This phase also served to show participants how *list questions* are formulated. The results from this phase indicate that our classifier activates reliably for known questions being asked, often on the first attempt.

The informed exploratory phase tested whether participants could formulate correct *list questions* using their understanding of the classifier’s scope, given the two previous phases. Interviewers provided participants with conversation topics, and asked them to formulate a unique question for each topic. The results from this phase show that participants were successful at formulating new *list questions*. Additionally, this shows that the classifier is easy to understand after a brief explanation and a few examples.

Overall, the activation rate for the classifier, shown in Figure 30b, increased significantly between the uninformed and informed exploratory phases. These results are promising and show that the classifier can be activated by many individuals with little knowledge of the application.

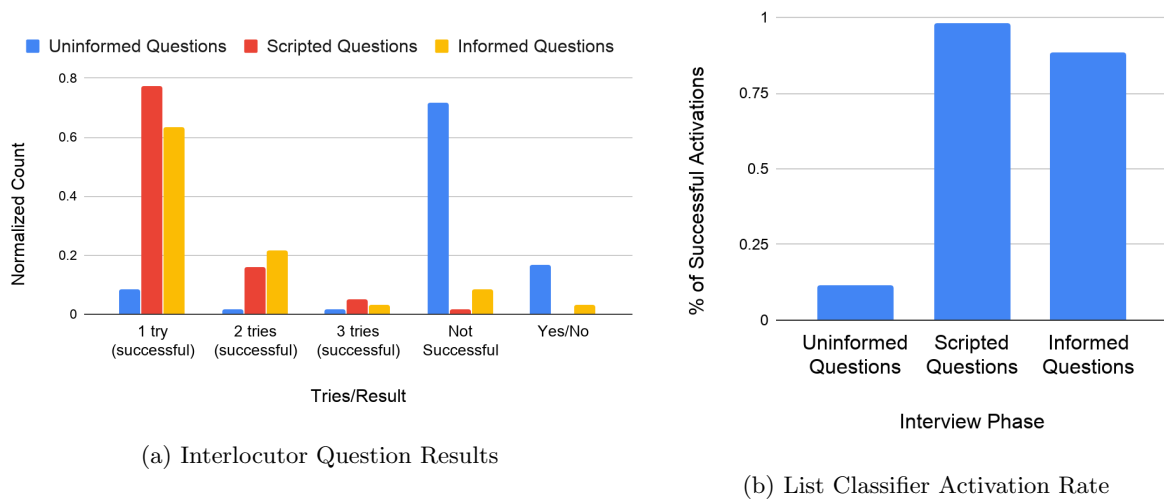
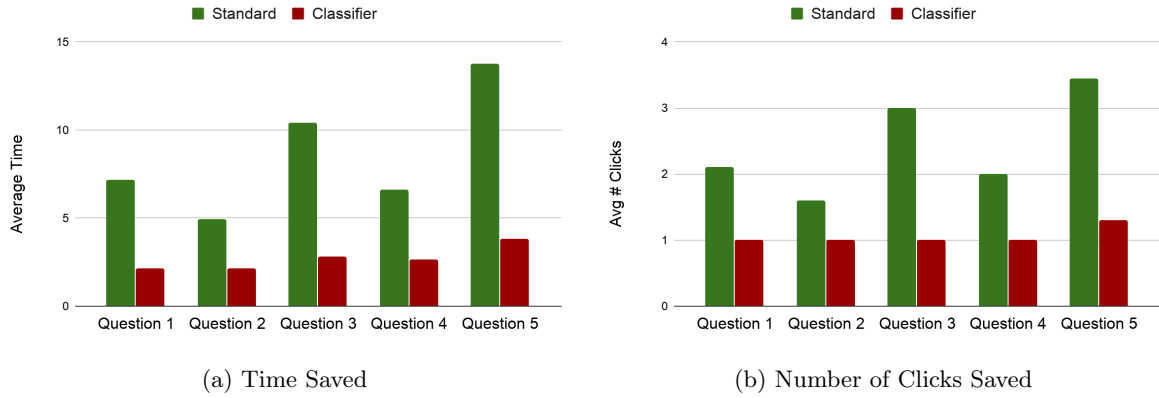


Figure 30: Interlocutor Set Results

There were several limitations on the *interlocutor set* of our interviews. In early studies conducted, we noted the tablet’s microphone had poor accuracy with the included Android Speech Recognition library. This meant that questions in the scripted phase, which should all be successful, were not recorded properly and could never be recognized by the classifier. In order to overcome this we used an inexpensive lapel microphone that we asked participants to wear. This greatly improved the results of the Android Speech Recognition library, and similarly presented more accurate questions to the classifier. Additionally, we found a bug in our code that mistakenly classified *non-list questions* as *list questions* at regular intervals. Therefore, some of the activations recorded in Figure 30a are false positives that would have otherwise not activated the classifier. This bug was identified and fixed in early interviews, resulting in no false positive activations in later tests. Each of these limitations negatively impacted our results, however neither was a limitation of our feature’s design. We believe if we had fixed these issues before we began our study, our results would be even better, showing just how powerful the classifier design is.

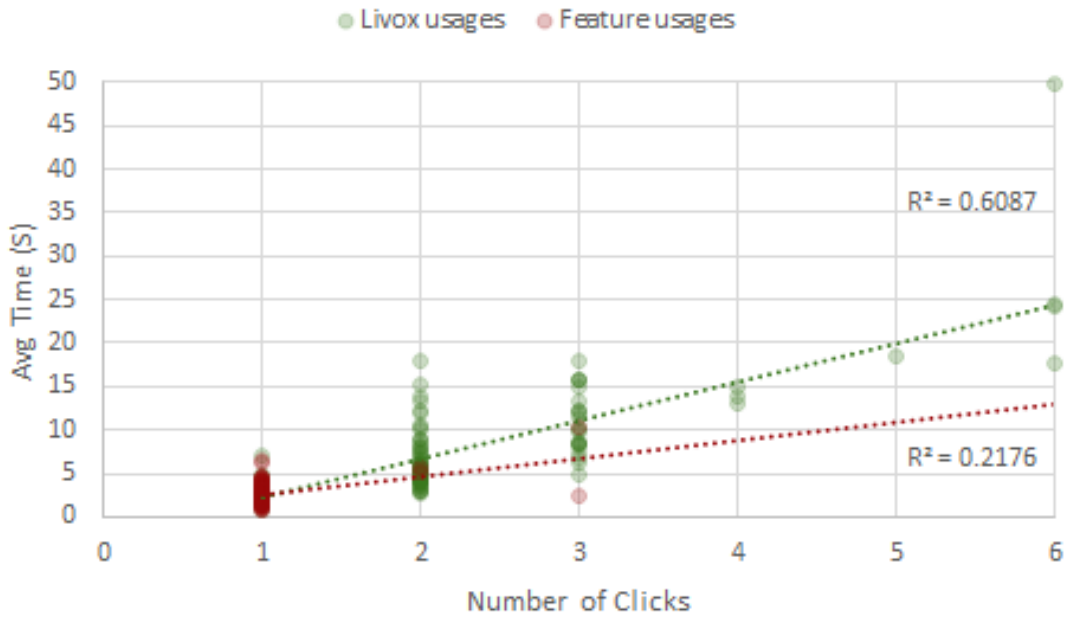
7.4.2 User Interview Results

The results from the user interview phase show that our classifier significantly reduces the time and effort to communicate through the Livox application. The first phase of this interview had participants use the Livox application, with our classifier disabled, to measure the baseline for the time and effort required to answer questions in conversation. The second phase of the interview had participants answer the same questions in conversation, but this time with our classifier enabled. For each phase we recorded the time and number of items pressed to respond to the question.



(a) Time Saved

(b) Number of Clicks Saved



(c) Avg Time by number of clicks

Figure 31: User Study Results

We combined the results from these phases, shown in Figure 31a and Figure 31b, to highlight the comparative time and effort required to answer each question. The results show that far less time and fewer clicks were required to answer all questions through the list classifier. Additionally, there is a clear correlation between the time and number of clicks required to find a response in the standard application, as seen in Figure 31c. For example, the final question asked in each phase took both the greatest amount of time and clicks to respond. Similarly, we found that when responses were nested in multiple folders within the application, the time and effort required greatly increased. Although we did not test the application and feature with any questions requiring the participant to navigate more than three folders deep to find a response, the number of clicks would continue to increase if responses were nested even further within the application. We similarly found that when responses had to be located in menus with many response options, the time to locate the response greatly increased even though the participant did not have to navigate nested menus.

Overall the time and effort required to respond to questions was greatly reduced through using our classifier. The total time and number of clicks required to answer all questions, shown in Figure 32a and Figure 32b, also indicates that the savings are compounded over many questions being asked during a conversation. Participants required less than half the number of clicks, and just over a quarter of the time to answer all questions through the list classifier compared to the standard application. Based on this, it is clear that our classifier greatly reduces the time and effort required to communicate

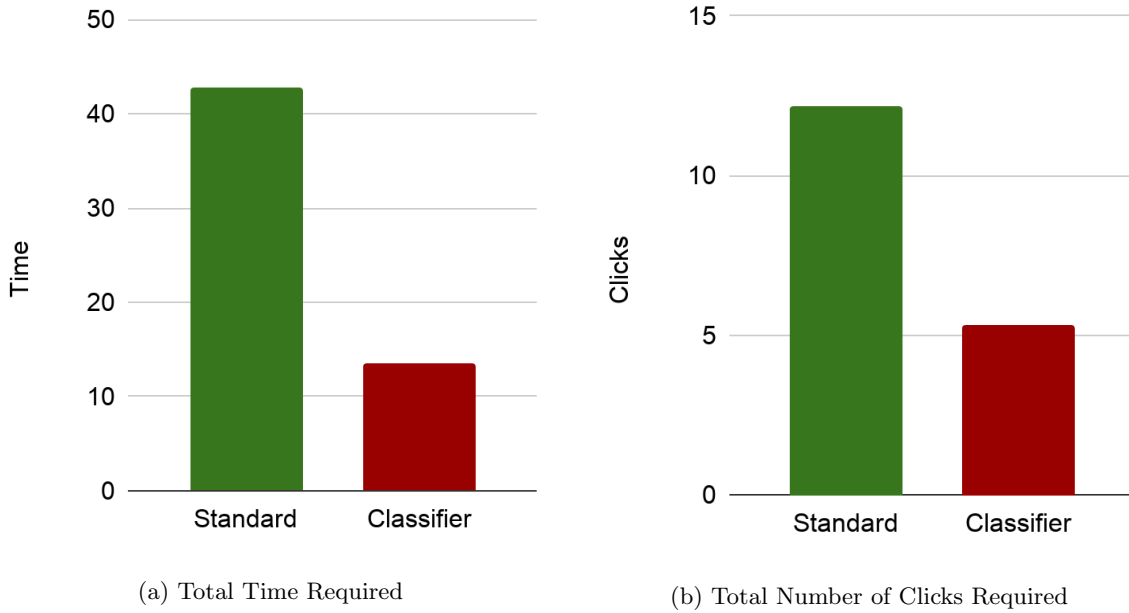


Figure 32: Resources Used

while using the Livox application in conversation.

We noted that our interviews were conducted only with individuals without disabilities, for which we had increased the number of visible items on the screen to a best-case scenario. Many of Livox’s users have fewer items presented on each screen, causing some items to fall on successive pages, requiring additional clicks or swiping between pages at the same folder level. We believe that reducing the number of visible items, as is more common with Livox’s users, would further increase the time and number of clicks needed to respond to questions. Under these scenarios our classifier would have an even greater impact on facilitating conversations.

7.4.3 Exploratory Question Results

We concluded each interview with a set of exploratory questions to collect participant’s feedback on our *list question* classifier and the Livox application in general. All participants reported to have a positive experience using the application. In early interviews, before we fixed some reliability bugs and before we found an accurate microphone, many respondents voiced concerns about the reliability of our voice-activated feature. Although some of these concerns were alleviated with the bug fixes we implemented, we still noticed poor reliability of the feature’s activation due to microphone quality and the results produced by the Android speech recognition library. These challenges were magnified when using the application in noisy or crowded environments with many people speaking at the same time.

Because these challenges faced were not originally encapsulated in the goals of this project, we did not put much time into their solution. Our team believed our time was best spent improving the *list question* classifier given the limitations of current speech recognition libraries. We hope that the reliability of our feature increases with further improvements to the Android speech recognition library.

8 Discussion and Related Work

Although the results of our evaluation show that our classifier activates reliably and presents relevant responses for most questions, the classifier can still be improved by applying different strategies to each service it provides. In this section, we suggest improvements that can be made to each component of our solution: question recognition, phrase parsing, entity parsing, and image retrieval. Additionally, We suggest several ways in which the classifier can be better integrated into the Livox application.

Early in our project development, we linked our online API to a logging database. This recorded each question sent to our API, a flag whether it is a *list question* or not, and the split *question* and *response phrases*. While testing the classifier in the Livox application during day-to-day research and user studies, we collected over 800 total questions. Many of these questions are repeats, as we often tested the API using the same questions for debugging purposes. However, there are many unique questions, including some *yes or no* questions that were not filtered by Livox’s existing classifier. We believe this logged data could be used to identify weaknesses in classifier recognition, to better tune the heuristic solutions, and used as training data for future machine learning models.

8.1 Question Recognition Improvements

Although our initial machine learning solution for *list question* recognition did not perform as well as our later heuristic solution, we believe it would be worthwhile to develop another machine learning solution to achieve this. This model could be trained in part using the data collected through our API logs along with other question datasets, such as the Amazon product dataset used for our initial model. We believe a more powerful machine learning solution might generalize better for recognizing open-ended questions, which we discuss later in 8.4 Entity Parsing Improvements.

8.2 Phrase Parsing Improvements

Our phrase parsing solution, although effective for most questions asked, can be improved upon. The offsets used to split *question* and *response phrases* occasionally remove important information from the *response phrase*, causing the application to omit relevant responses. However, because the parser estimates conservatively, it is more common that extraneous responses are presented. There are several ways to improve these issues. Our parser decides the offsets using a predefined dictionary of common *question words* and associated prepositional phrases, the values of which were chosen to be conservative rather than based on data. The dictionary words and offset values could be further tuned using insights from logged data and results from our crowdsourced dataset. Alternatively, the combination of our logged data and crowdsourced dataset could be augmented and used to train a machine learning model for this purpose. A machine learning solution would likely be scalable and account for uncommon questions and *response phrases*.

8.3 Entity Parsing Improvements

Our entity parsing solution was proven to be strong based on the results from our crowdsourced testing questions. However, we believe this can be improved. Our categorized results show that the classifier is least accurate for responses including *organizations*, *numbers*, and *people*. The *organization* category can be improved by adding lists of common organizations and household brands to our vocabulary of recognizable entities. Person recognition could be improved by adding lists of celebrities and user-generated lists of people of whom the user interacts with in daily life.

One study participant suggested, during our final exploratory question phase, presenting users with supplemental response options in the conversation category of the question being asked. For example, when asking “what would you like for dinner: pizza or pasta” the classifier would display

additional response options for dinner items not mentioned. This functionality would be beneficial to users in the case that their desired choice was not one of the response options presented, but is very similar.

Our team identified two ways in which this could be achieved. The simpler solution identified is to expand the presented options by including similar responses from the word embedding model used to derive our vocabulary. Our classifier would for example, identify a predefined number of responses with the highest cosine similarity to each of the successfully recognized entities in the *response phrase*.

The more difficult solution our team identified is to classify the conversation topic of a question as it is being processed. The same process as before can then be applied to find a predefined number of responses similar to this category in our word embedding model. We believe this solution would be possible through the use of a machine learning model or a semantic analysis dependency tree. The model could be trained, in part, by our crowdsourced dataset of questions, as we instructed respondents to label each question with its conversation topic. We believe the ideal solution would be a combination of the two proposed here. By calculating similarity compared to the extracted entities, our classifier would be more likely to present relevant options. However, in the case that our question recognition module is improved with a machine learning solution to recognize open-ended questions, our classifier could present generic options for the conversation category being asked about. For example, if an interlocutor asked “what would you like for lunch: a hamburger or a hotdog,” our classifier could present similar responses, such as pizza. At the same time, if an interlocutor asked generically “what would you like for lunch,” our classifier could present generic response options for lunch foods.

8.4 Image Retrieval Improvements

There are several ways in which our image retrieval process could be improved as well. We believe an important addition to our online API and our classifier as a whole, is to develop a solution to label new images being imported into the Livox application. Currently, users and caregivers can create, upload, and download original content through the online Livox Portal. This downloaded content can be imported into the application, but cannot be found by our classifier, as none of the images will have our custom labels. Labels can be dynamically built through multiple queries to the Google Vision API for each new downloaded image. Additionally, for small packages of downloaded content, it would be reasonable to allow users or caregivers to manually label each image in the style of our vocabulary. This functionality would, in general, enable users and their families to use custom images within this natural conversation tool to display ideas not currently supported by the Livox application.

Our image relevance scoring results show that errors in image retrieval are most common for *numbers*, *organizations*, *date/times*, and *miscellaneous* entities. For the cases where no image is found for a given response, the method described above could be used to insert additional images into the Livox application to support those ideas or responses. For the cases where an irrelevant image is found, there are many little improvements that can be made. For example, researchers can manually modify the labels and associated confidence scores for specific images known to be irrelevant. Users or caregivers could similarly be able to manually identify irrelevant images when they appear, as to ensure they are not presented again.

Image retrieval could be further improved by querying our database of images with the question’s conversation category in addition to the response being related. For example, in Figure 23, the image for “Score of 2” would be relevant if the conversation category were *activities*, however the image is not relevant for the category *date/time*. Again, this relies on the ability to classify a question’s topic while it is being processed. This addition would benefit our image retrieval for common responses, like the example “fall,” which can be used in many contexts with different meanings.

8.5 Livox Integration Improvements

Many user study participants suggested ways in which our classifier could be better integrated into the Livox application. Our current integration loads a new screen when questions were detected

that only presented tiles for the response options recognized. One participant suggested presenting the items on the same, main screen as the rest of the standard icons. This would allow users to take advantage of the classifier if it presented relevant options, however users would still have the ability to navigate the application as standard if they wished to find an alternative answer to the question. Another participant suggested adding visual feedback to the user interface showing whether the classifier recognized any speech at all. Currently there is no feedback showing whether the question was not heard properly, or whether there was another error in classifying the question. This would at least show users whether their questions are being improperly formulated to be recognized, or whether there are unrelated issues in speech recognition. We believe this visual feedback would improve the activation rate of our classifier.

9 Conclusion

The deliverable of this project is a voice activated, NLP-based classifier, within the Livox application, that facilitates natural conversation for people with disabilities. This classifier detects open ended questions, followed by a list of response options, and presents relevant pictograms to users. The classifier uses lightweight processes and runs well on resource constrained devices, while also working in offline environments. It is intuitive and easy to use while also keeping a high level of accuracy for common speech.

The feature was shown to be effective by individually evaluating each step in our pipeline. The interlocutor set of the user study evaluated the performance of classifying questions in real world situations, giving an 88.3% accuracy with participant-created questions. The crowdsourced dataset enabled us to evaluate the phrase parsing, and entity parsing processes. We found that 90% of sentences were split by the phrase parser such that all responses were included in the *response phrase*. In addition, entities were correctly extracted from 63.3% of *response phrases*. Most entity extraction failures were not due to missing entities, but rather compound nouns and phrases that were not combined correctly into single entities. Finally, the image retrieval process matched relevant images for entities 68.3% of the time and returned no image, rather than an irrelevant image, 13.4% of the time.

Most importantly, our added feature has proven to reduce the reciprocity gap in conversation by making communication faster and easier. Results from the user study showed that there was an average reduction of 56.4% in the number of clicks needed to answer questions. We also found a reduction of 68.5% in the time required for users to respond compared to the standard application. This feature serves as a practical example of how voice-activated functionality can assist people with verbal and motor challenges, and is a step towards ensuring that verbal and motor challenges can be overcome by applying state of the art technologies in a human-centric implementation.

Works Cited

- [1] Division for Inclusive Social Development, “Realization of the sustainable development goals by, for, and with persons with disabilities,” United Nations Dept. of Social and Economic Affairs, report, 2018. [Online]. Available: <https://www.un.org/development/desa/disabilities/wp-content/uploads/sites/15/2018/12/UN-Flagship-Report-Disability.pdf> (visited on 01/2020).
- [2] Center for Disease Control and Prevention. (2020). 11 things to know about cerebral palsy, Center for Disease Control and Prevention, [Online]. Available: <https://www.cdc.gov/features/cerebral-palsy-11-things/index.html> (visited on 10/2019).
- [3] S. Mizunoya, S. Mitra, and I. Yamasaki, “Towards inclusive education: The impact of disability on school attendance in developing countries,” *Innocenti Working Paper*, vol. 2016, 3 May 23, 2016. DOI: 10.2139/ssrn.2782430. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2782430# (visited on 10/2019).
- [4] American Speech-Language-Hearing Association. (2019). Information for aac users, [Online]. Available: <https://www.asha.org/public/speech/disorders/Information-for-AAC-Users/> (visited on 10/2019).
- [5] K. T. Chazin, E. D. Quinn, and J. R. Ledford. (2016). Augmentative and alternative communication (aac), [Online]. Available: <http://ebip.vkcsites.org/augmentative-and-alternative-communication> (visited on 10/2019).
- [6] A. S. Bondy and L. A. Frost, “The picture exchange communication system,” *Behavior Modification*, vol. 25, pp. 725–744, 5 2001. [Online]. Available: <https://journals.sagepub.com/doi/abs/10.1177/108835769400900301> (visited on 10/2019).
- [7] Autism Connection of Pennsylvania. (2013). Autism tool kit, Autism Connection of Pennsylvania, [Online]. Available: <http://autism-support.org/autism-resources/autism-tool-kit/> (visited on 10/2019).
- [8] L. Collet-Klingenberg. (2008). Pecs: Steps for implementation, The National Professional Development Center on Autism Spectrum Disorders, [Online]. Available: https://autismpdc.fpg.unc.edu/sites/autismpdc.fpg.unc.edu/files/PECS_Steps.pdf (visited on 10/2019).
- [9] Pyramid Educational Consultants. (2020). Small communication book, Pyramid Educational Consultants, Inc., [Online]. Available: <https://pecsaustralia.com/shop/small-communication-book/> (visited on 03/2020).
- [10] AliMed. (2019). Gotalk express 32, [Online]. Available: <https://www.alimed.com/gotalk-express-32.html> (visited on 10/2019).
- [11] PRC-Salttillo. (2020). What is aac? PRC-Salttillo, [Online]. Available: <https://www.prentrom.com/caregivers/what-is-augmentative-and-alternative-communication-aac> (visited on 10/2019).
- [12] AbleNet. (2020). Quicktalker 7, [Online]. Available: <https://www.ablenetinc.com/technology/speech-generating-devices/quicktalker-7> (visited on 10/2019).
- [13] M. Rodríguez-Fórtiz, J. González, A. Fernández, M. Entrena, M. Hornos, A. Pérez, A. Carrillo, and L. Barragán, “Sc@ut: Developing adapted communicators for special education,” *Procedia - Social and Behavioral Sciences*, vol. 1, no. 1, pp. 1348–1352, 2009, World Conference on Educational Sciences: New Trends and Issues in Educational Sciences, ISSN: 1877-0428. DOI: <https://doi.org/10.1016/j.sbspro.2009.01.238>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1877042809002419>.
- [14] H. Schelhowe and S. Zare, “Intelligent mobile interaction: A learning system for mentally disabled people (imlis),” *Stephanidis C. (eds) Universal Access in Human-Computer Interaction. Addressing Diversity. UAHCI 2009. Lecture Notes in Computer Science*, vol. 5614, pp. 234–243, 2009. DOI: 10.1007/978-3-642-02707-9_47. (visited on 10/2019).
- [15] AssistiveWare B.V. (2020). Proloquo2go, AssistiveWare B.V., [Online]. Available: <https://www.assistiveware.com/products/proloquo2go> (visited on 10/2019).
- [16] T. Dynavox. (2020). Snap for windows, [Online]. Available: <https://www.tobiidynavox.com/en-us/software/windows-software/snap-for-windows/> (visited on 10/2019).

- [17] Gus Communication Devices Inc. (2020). Talktablet, Gus Communication Devices Inc., [Online]. Available: <https://talktablet.com/> (visited on 10/2019).
- [18] AssistiveWare. (2020). Progressive language, [Online]. Available: <https://www.assistiveware.com/innovations/progressive-language> (visited on 10/2019).
- [19] P. Chandrayan. (Oct. 22, 2017). A guide to nlp : A confluence of ai and linguistics, [Online]. Available: <https://codeburst.io/a-guide-to-nlp-a-confluence-of-ai-and-linguistics-2786c56c0749> (visited on 10/2019).
- [20] S. M. Palakollu. (Aug. 31, 2019). Top 5 natural language processing python libraries for data scientist, Towards Data Science, [Online]. Available: <https://towardsdatascience.com/top-5-natural-language-processing-python-libraries-for-data-scientist-32463d36feae> (visited on 10/2019).
- [21] C. Cherpas, “Natural language processing, pragmatics, and verbal behavior,” *The Analysis of verbal behavior*, vol. 10, pp. 135–147, 1992. DOI: 10.1007/bf03392880. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pubmed/22477052> (visited on 12/2019).
- [22] H. Zulkifli. (Aug. 26, 2018). Linguistic knowledge in natural language processing, Towards Data Science, [Online]. Available: <https://towardsdatascience.com/linguistic-knowledge-in-natural-language-processing-332630f43ce1> (visited on 10/2019).
- [23] D. Sarkar. (Jun. 19, 2018). A practitioner’s guide to natural language processing (part i) — processing & understanding text, Towards Data Science, [Online]. Available: <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72> (visited on 10/2019).
- [24] W. Phillips. (2006). Introduction to natural language processing, Consortium on Cognitive Science Instruction, [Online]. Available: http://www.mind.ilstu.edu/curriculum/protothinker/natural_language_processing.php (visited on 12/2019).
- [25] K. Krishnan. (May 21, 2019). How do alexa skills work? [Online]. Available: <https://chatbotsmagazine.com/how-does-alexa-skills-works-82a7e93dea04> (visited on 10/2019).
- [26] Amazon.com, Inc. (2019). Amazon echo, Amazon.com, Inc., [Online]. Available: <https://www.amazon.com/all-new-amazon-echo-speaker-with-wifi-alexa-dark-charcoal/dp/B06XCM9LJ4> (visited on 10/2019).
- [27] Google. (2019). Google home - smart speaker and home assistant - google store, Google, [Online]. Available: https://store.google.com/us/product/google_home?hl=en-US (visited on 10/2019).
- [28] B. Vigliarolo. (Sep. 27, 2019). Alexa skills: Cheat sheet, [Online]. Available: <https://www.techrepublic.com/article/alexa-skills-cheat-sheet/> (visited on 10/2019).
- [29] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008, ch. 1.2.4-6.2.1, ISBN: 0521865719. [Online]. Available: <https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html>.
- [30] W. Scott. (Feb. 15, 2019). Tf-idf from scratch in python on real world dataset, Towards Data Science, [Online]. Available: <https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089> (visited on 12/2019).
- [31] C. Sutton and A. McCallum, *An Introduction to Conditional Random Fields*. now, 2012. DOI: 10.1561/22000000013. [Online]. Available: <https://ieeexplore.ieee.org/document/8186901>.
- [32] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [33] W. D. Travis, E. Brambilla, A. G. Nicholson, Y. Yatabe, J. H. Austin, M. B. Beasley, L. R. Chirieac, S. Dacic, E. Duhig, D. B. Flieder, K. Geisinger, F. R. Hirsch, Y. Ishikawa, K. M. Kerr, M. Noguchi, G. Pelosi, C. A. Powell, M. S. Tsao, and I. Wistuba, “The 2015 world health organization classification of lung tumors: Impact of genetic, clinical and radiologic advances since the 2004 classification,” *Journal of Thoracic Oncology*, vol. 10, no. 9, pp. 1243–1260, 2015, ISSN: 1556-0864. DOI: <https://doi.org/10.1097/JTO.0000000000000630>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1556086415335711>.
- [34] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of CNN and RNN for natural language processing,” *CoRR*, vol. abs/1702.01923, 2017. arXiv: 1702.01923. [Online]. Available: <http://arxiv.org/abs/1702.01923>.

- [35] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," English, *IET Conference Proceedings*, 850–855(5), Jan. 1999. [Online]. Available: https://digital-library.theiet.org/content/conferences/10.1049/cp_19991218.
- [36] A. Mansouri, L. Affendey, and A. Mamat, "Named entity recognition approaches," *International Journal of Computer Science and Network Security*, vol. 8, pp. 339–344, 2 Jan. 2008. [Online]. Available: https://www.researchgate.net/publication/238607553_Named_Entity_Recognition_Approaches.
- [37] Stanford NLP Group. (Oct. 16, 2018). Stanford named entity recognizer (ner). version 3.9.2, The Stanford NLP Group, [Online]. Available: <https://nlp.stanford.edu/software/CRF-NER.html>.
- [38] Explosion AI. (2019). Spacy: Industrial-strength natural language processing, Explosion AI, [Online]. Available: <https://spacy.io/> (visited on 10/2019).
- [39] NLTK Project. (2019). Natural language toolkit, NLTK Project, [Online]. Available: <https://www.nltk.org/> (visited on 03/2019).
- [40] The University of Sheffield. (2019). General architecture for text engineering, [Online]. Available: <https://gate.ac.uk/> (visited on 03/2019).
- [41] DataTurks. (May 2, 2018). Stanford corenlp: Training your own custom ner tagger., medium.com, [Online]. Available: <https://medium.com/swlh/stanford-corenlp-training-your-own-custom-ner-tagger-8119cc7dfc06> (visited on 10/2019).
- [42] R. Chawla. (Aug. 7, 2017). Overview of conditional random fields, medium.com, [Online]. Available: <https://medium.com/ml2vec/overview-of-conditional-random-fields-68a2a20fa541> (visited on 10/2019).
- [43] K. M. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. J. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2013.
- [44] Google. (2019). Google drive, Google, [Online]. Available: <https://www.google.com/drive/> (visited on 08/2019).
- [45] Atlassian. (2019). Jira issue and project tracking software, Atlassian, [Online]. Available: <https://www.atlassian.com/software/jira> (visited on 09/2019).
- [46] Slack. (2019). Slack, Slack, [Online]. Available: <https://slack.com/> (visited on 10/2019).
- [47] WhatsApp Inc. (2019). Whatsapp, WhatsApp Inc., [Online]. Available: <https://www.whatsapp.com/> (visited on 10/2019).
- [48] GitHub, Inc. (2019). Github, GitHub, Inc., [Online]. Available: <https://github.com/> (visited on 10/2019).
- [49] Google Developers. (2019). Android studio, Google, [Online]. Available: <https://developer.android.com/studio> (visited on 08/2019).
- [50] JetBrains s.r.o. (2019). Pycharm, JetBrains s.r.o., [Online]. Available: <https://www.jetbrains.com/pycharm/> (visited on 08/2019).
- [51] Pallets Projects. (2019). Flask. version 1.1.1, The Pallets Projects, [Online]. Available: <https://palletsprojects.com/p/flask/> (visited on 08/2019).
- [52] Amazon.com, Inc. (2019). Amazon web services (aws), Amazon.com, Inc, [Online]. Available: <https://aws.amazon.com/> (visited on 08/2019).
- [53] Google. (2019). Google cloud, Google, [Online]. Available: <https://cloud.google.com/> (visited on 08/2019).
- [54] Microsoft Corporation. (2019). Microsoft azure, Microsoft Corporation, [Online]. Available: <https://azure.microsoft.com/en-us/> (visited on 08/2019).
- [55] Amazon.com, Inc. (2005). Amazon mechanical turk, Amazon.com, Inc, [Online]. Available: <https://www.mturk.com/> (visited on 01/2020).
- [56] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, Nov. 2016, pp. 44–51. DOI: 10.1109/SOCA.2016.15.

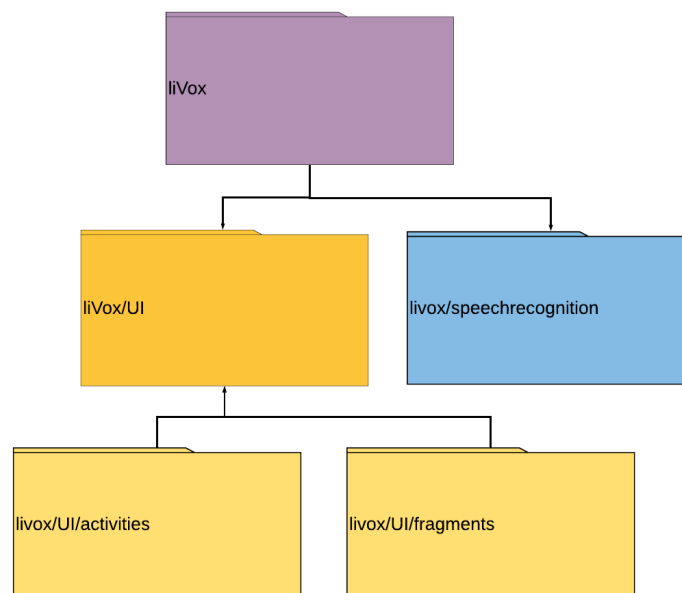
- [57] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, “BoolQ: Exploring the surprising difficulty of natural yes/no questions,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 2924–2936. DOI: 10.18653/v1/N19-1300. [Online]. Available: <https://www.aclweb.org/anthology/N19-1300>.
- [58] UX For The Masses. (May 15, 2018). How to run an exploratory ux testing session, [Online]. Available: <http://www.uxforthemasses.com/exploratory-ux-testing> (visited on 10/2019).
- [59] J. McAuley. (2020). Amazon question/answer data, [Online]. Available: <http://jmcauley.ucsd.edu/data/amazon/qa/> (visited on 11/2019).

Appendix A: Livox Code Analysis

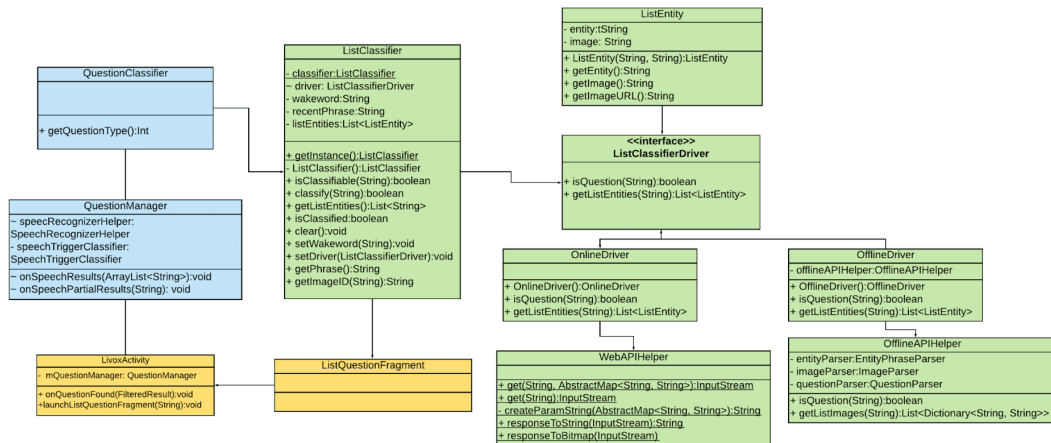
The Livox codebase, managed through bitbucket, originally had very limited documentation explaining the structure of code and flow of data within the application. We started annotating and creating documentation for the Livox application's codebase, making the process of integrating any features into the application easier, since we know exactly how each of the packages fit together, and where we will add code to avoid disrupting existing features of the application. In this section, we will discuss the general architecture of the application, as well as the architecture within the specific packages for speech recognition and for machine learning.

General Architecture

The Livox codebase is standard for Android applications, using Java and Kotlin for back-end control, and XML for front end views. The Livox application is designed based on the entity, boundary, controller (EBC) architectural style. This style was developed with four main concerns in mind: data modeling and persistence, code modularity, extensibility, and testing. The codebase is broken into classes based on the independent pieces of functionality that the application requires. Entity classes are identified and created in order to maintain consistent and reliable state within the application. Entity classes are designed to represent, as accurately as possible, the real world entities that fall in the application's domain. Boundary classes are static interfaces that handle the direct interaction between the user and the application. Boundary classes do not maintain any state of the system or handle any logic of the application. Controller classes hold all of the logic of the application and implement all of the use cases that users may wish to complete. Controller classes manipulate the front-end boundary classes as well as modify the state of the application, so long as it is a discrete transformation from one steady-state to another.



This categorization and separation of classes enables the modularity, extensibility, and testing design goals. Since each use case of the application is likely to get its own controller or package of controller classes, it is easy to implement new use cases with relatively little modification to the existing code. Separation of classes also enables easier testing of the application, since all of the formal logic involved in the application is encapsulated in the controller classes, which can be tested independently of the static boundary objects.



Speech Recognition

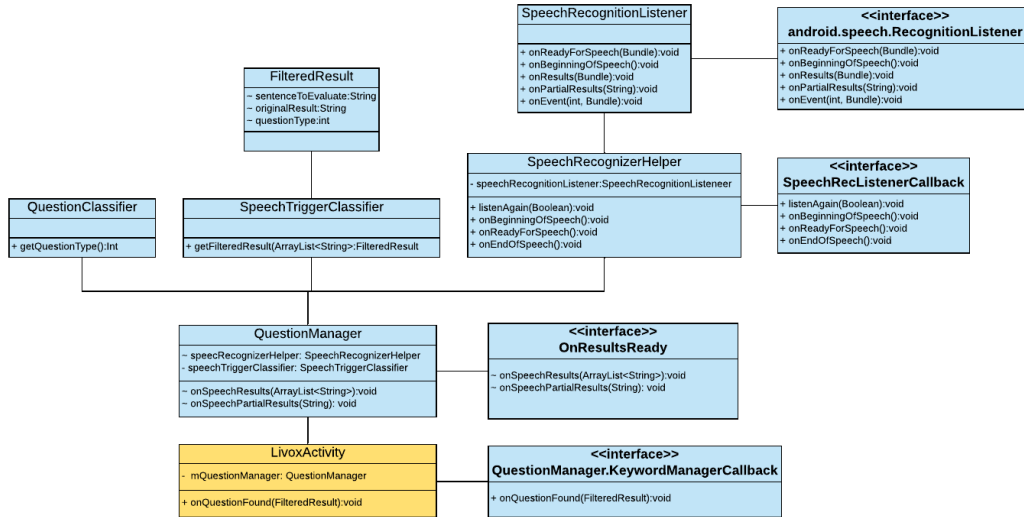
The Livox application encapsulates all control for speech recognition in an independent package of classes. The class diagram in Figure packarch shows the architecture of this package. The package has separate controller classes for different steps in the data pipeline: speech-to-text, text normalization, sentence segmentation, and phrase classification. This separation between preprocessing and classification enables porting of the package to a standalone application, and modification of specific steps in the pipeline through minimal refactoring of code. When we reintegrate this code, we will be able to insert our own sentence segmentation without disturbing any existing code for speech-to-text, text normalization, or phrase classification.

Figure packarch. Livox Speech Recognition Package Architecture

This package primarily uses callback functions, defined at appropriate levels of the architecture, to pass data up the pipeline of controller classes. The controller for coordinating phrase classification delegates to classes in the machine learning package in order to classify individual pieces of a given question. The highest-level controller class then uses a callback to vb the main UI class which coordinates launching the appropriate screen to respond to the question.

Machine Learning and Classifiers

The Livox application similarly encapsulates all controllers for machine learning models in a separate package. The majority of classes in this package are used to generate recommended items for the user based on time, location, and pattern usage. There are, however, some classes in this package with models used to classify questions recognizable by the application. The phrase classification controller class in the speech recognition package directly interacts with these classes. These machine learning based phrase classification controllers also interface with the Android WEKA package's classifier and local data utilities classes. We will not implement our own machine learning models for phrase classification in this package, since the package handles only binary question classification and the overwhelming majority of classes are used for identifying recommended items. We will create a separate package for phrase classification that encapsulates its own machine learning models. We will design the controllers in this package based on a similar architecture to the existing binary classifier.



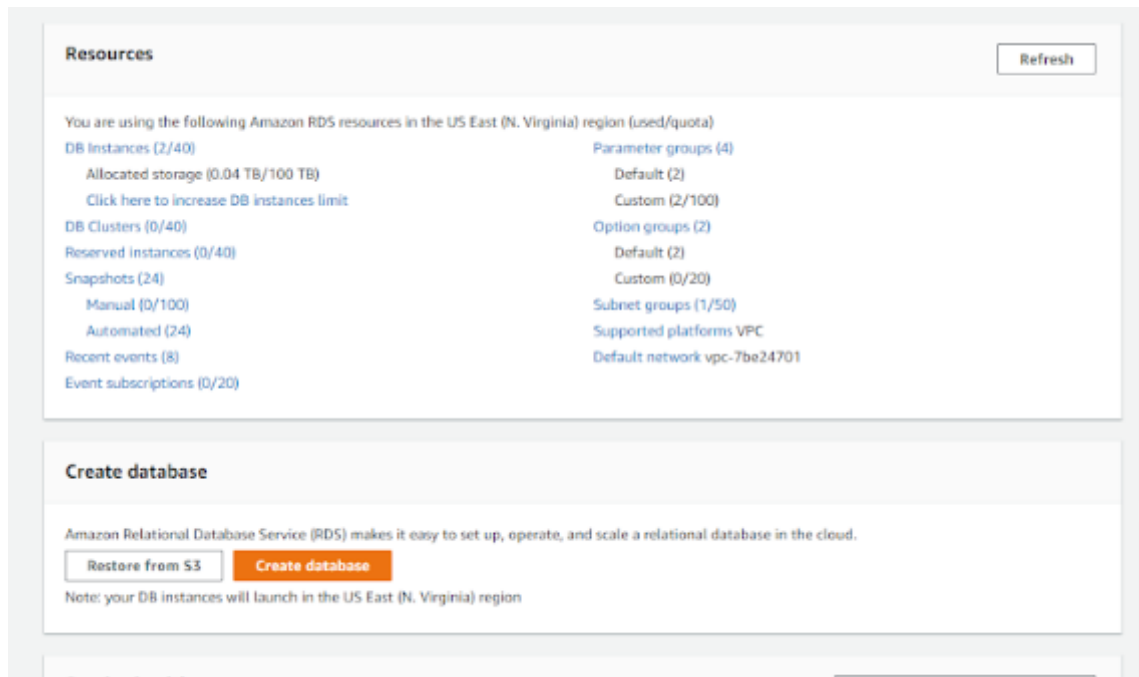
Given the existing architecture in the Livox codebase, it will be easier to expand on the speech recognition package with our own preprocessing, and create our own package for question classification. This ease-of-extensibility is a direct result of EBC architecture that was originally used when creating the application, and will be maintained as new features are added to the codebase.

Appendix B: Online Classifier Deployment

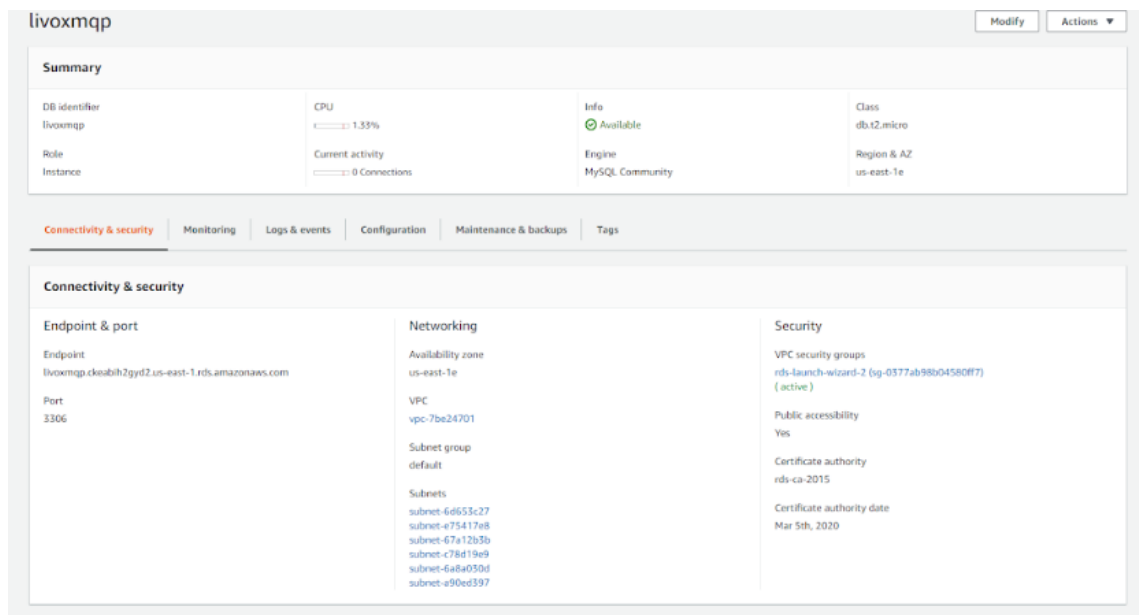
The following documentation is for the purpose of deploying the Online Classifier on AWS. The code for our online classifier can be found at this Github repository. <https://github.com/rcvalenteai/livox-online-classifier>

Deploy Database on Amazon RDS

First create a new AWS RDS Database Instance. A simple tutorial to achieve this can be found here: [Creating an RDS Instance](#)

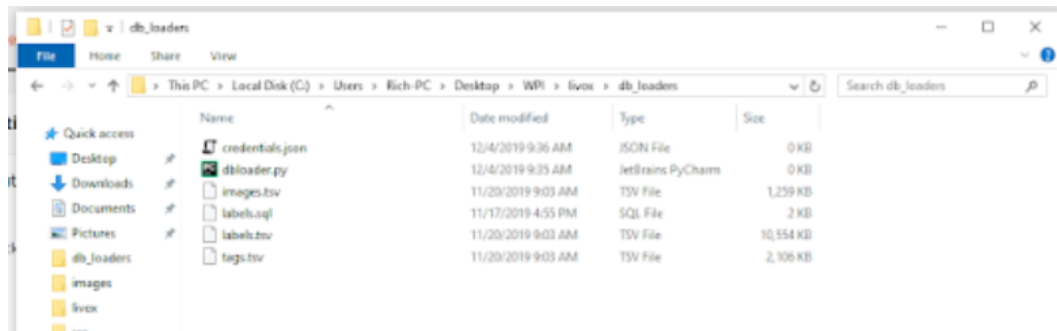


Second, find your connection credential information of your instance under the 'Connectivity & security' tab. Find the **DB Identifier**, **endpoint**, and **password**.



Load Data into the Database

Navigate to the “/src/db-loaders” directory in the provided git repository. The next steps will use files in the db-loaders folder.



Edit the “credentials.json” file with the following information from Step 1

```
{
user: enter your DB User,
password: enter your DB password,
host: enter the rout to your RDS instance
}
```

Open a terminal to this folder, and run the “dbloader.py” script using python

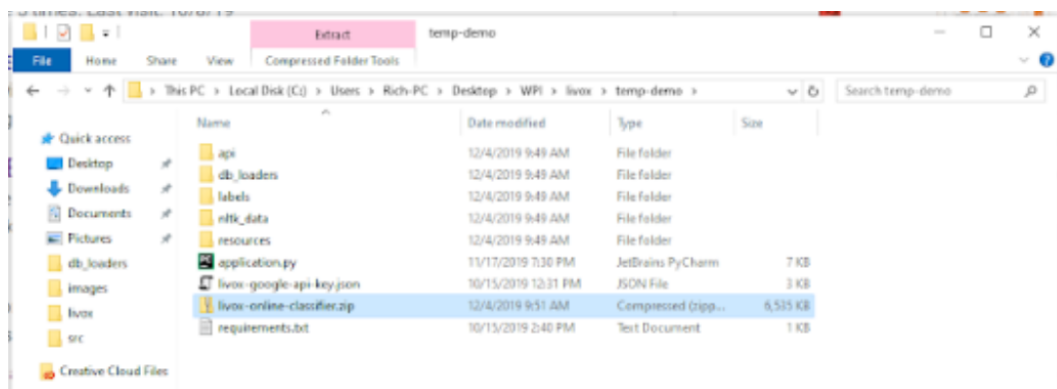
```
>>> python dbloader.py
```

This will initialize the AWS database.

Deploy Flask API to Elastic Beanstalk

Next, deploy the Flask Application on Elastic Beanstalk. An in depth tutorial on starting an Elastic Beanstalk instance can be found here: [Deploying Flask App to Elastic Beanstalk](#)

Once the Elastic Beanstalk Instance is deployed, Zip up the contents of the src folder to be uploaded.



In order to deploy the Zipped API code:

1. Navigate to your Elastic Beanstalk deployment server, click upload and deploy
2. Upload your zipped folder
3. Done!

The screenshot shows the AWS Elastic Beanstalk console for an application named 'api-server' in the 'ApiServer-env' environment. The interface includes a left-hand navigation menu with options like Dashboard, Configuration, Logs, Health, Monitoring, Alarms, Managed Updates, Events, and Tags. The main content area is titled 'Overview' and features several key metrics: 'Health' is 'Ok' with a green checkmark icon and a 'Causes' button; 'Running Version' is 'Ivoox v1-33' with an 'Upload and Deploy' button; and 'Platform' is 'Python 3.6 running on 64bit Amazon Linux/2.8.6' with a 'Change' button and a note that a 'Newer version available'. Below these metrics is a 'Recent Events' table with columns for Time, Type, and Details.

Time	Type	Details
2019-12-03 04:55:12 UTC-0500	INFO	Environment health has transitioned from Severe to Ok.
2019-12-03 04:53:12 UTC-0500	WARN	Environment health has transitioned from Ok to Severe. 100.0 % of the requests are erroing with HTTP 4xx.
2019-12-03 04:51:12 UTC-0500	INFO	Environment health has transitioned from Severe to Ok. 100.0 % of the requests are erroing with HTTP 4xx. Insufficient request rate (114.0 requests/min) to determine application health.
2019-12-03 04:49:12 UTC-0500	WARN	Environment health has transitioned from Ok to Severe. 100.0 % of the requests are erroing with HTTP 4xx.
2019-11-29 17:19:39 UTC-0500	INFO	Environment health has transitioned from Severe to Ok.

Appendix C: Livox Development Environment Setup

1. Clone the livox_android repo to a local development environment
2. Open the code repository in Android Studio
 - (a) new project → new project from version control → git → enter above url
3. Connect an Android device to run the App
 - (a) Physical Device: Enable USB debugging on the Android device and connect via a USB to your development machine
 - i. You may need to install Android File Transfer to access file storage on the device. (Should be MacOS computers only)
 - (b) Virtual Device: Open Android Virtual Device (AVD) manager and create a new virtual device. This should prompt you to download a virtual device driver.
4. Create a run configuration
 - (a) Note: If no modules are shown go to File → Project Structure → Modules → liVox and make sure the properties are correct
 - (b) Make sure the correct Android Device is selected in your run configuration
5. Run the application on your Android Device
 - (a) Possible Error: File google-services.json is missing. The Google Services Plugin cannot function without it.
 - i. The application needs to be connected with Firebase
 - ii. Create a firebase project here
 - iii. Under “Get started by adding Firebase to your app” click Android and download google-services.json,
 - iv. Add the downloaded JSON file to the ‘liVox’ package folder within the ‘livox_android’ repository
6. Sign in to Livox with an email address and installation key
 - (a) You can get installation keys from Carlos Pereira for development purposes.
7. Integration
 - (a) Livox used the ‘develop’ git branch for their own internal development
 - (b) We recommend branching off this branch when integrating any new features, and to pull and merge from it at regular intervals as to avoid versioning conflicts in the future.

Appendix D: IRB Approval Letter

Institutional Review Board FWA #00015024 - HHS #00007374

Notification of IRB Approval

Date: 27-Jan-2020

PI: Neamtu, Rodica

Protocol Number: IRB-20-0335

Protocol Title:

Towards a More Inclusive World: Enhanced Augmentative and Alternative Communication For People With Disabilities Using AI and NLP

Approved Study Personnel: Winsor, Cole;Robbertz, Andrew;Valente, Richard;Emil, Zachary;Neamtu, Rodica

Start Date: 27-Jan-2020 **Expiration Date:** 26-Jan-2021

Review Type: **Review Method:** Expedited Review **Risk Level:** Minimal Risk

Sponsor:

The WPI Institutional Review Board (IRB) approves the above-referenced research activity, having conducted a review according to the Code of Federal Regulations (45 CFR 46).

This approval is valid through 26-Jan-2021 unless terminated sooner (in writing) by yourself or the WPI IRB. Research activities involving human subjects may not continue past the expiration date listed above, unless you have applied for and received a renewal from this IRB.

We remind you to only use the stamped, approved consent form, and to give a copy of the signed consent form to each of your subjects. You are also required to store the signed consent forms in a secure location and retain them for a period of at least three years following the conclusion of your study. You are encouraged to use the InfoEd system for the storage of your consent forms.

Amendments or changes to the research must be submitted to the WPI IRB for review and approval before such changes are put into practice.

Investigators must immediately report to the IRB any adverse events or unanticipated problems involving risk to human participants.

Please contact the IRB at irb@wpi.edu if you have any questions.

*if blank, the IRB has not reviewed any funding proposal for this protocol

Appendix E: User Study Informed Consent

Informed Consent Agreement for Participation in a Research Study

Investigator:

Richard Valente, Tel. 1-908-566-6879, Email: rcvalente@wpi.edu
Zachary Emil, Tel. 1-202-999-7726, Email: zgemil@wpi.edu
Andrew Robbertz, Tel. 1-978-793-0506, Email: alrobbertz@wpi.edu
Cole Winsor, Tel. 1-978-844-0052, Email: ccwinsor@wpi.edu

Title of Research Study:

Towards a More Inclusive World: Enhanced Augmentative and Alternative Communication For People With Disabilities Using AI and NLP

Sponsors:

Carlos Pereira, Andre Camara

Introduction

You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study:

This project focuses on the creation of a voice activated feature within the Livox augmentative and alternative communication application. This feature enables the Livox application to listen for questions in conversation and present relevant responses to users. The purpose of this study is to evaluate the real world impact of this feature.

Procedures to be followed:

The following interview will take less than 30 minutes. The goal of this interview is to determine the accuracy of the new feature as well as to determine impact to users of the feature. First, you will be given tasks activating the new feature. These tasks will be conducted in three stages: exploratory questioning, scripted questioning, and informed exploratory questioning. Next, you will be given tasks to interact with the base Livox application. You will then repeat the tasks but interact using our feature. From using the application and our feature, we will gather data on the time and effort of using the application. The interview will be concluded with exploratory questions about the participants experience.

Risks to study participants:

The use of a tablet could lead to possible eye-strain, migraines, or epilepsy,

Benefits to research participants and others:

There are no direct benefits to the subjects in this study.

Record keeping and confidentiality:

With permission, we would like to take written and electronic records of the interview. Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators, the sponsor or its designee and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identify you by name. Any publication or presentation of the data will not

identify you.

Will you allow taking written and electronic records such as notes and audio recordings during the interview?

_Y _N Initials-----

Compensation or treatment in the event of injury:

There is minimal risk of injury or harm by participating in these interviews. Should you be injured during the interview, you will be cared for accordingly. You do not give up any of your legal rights by signing this statement.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact:

Investigators: Contact information above

IRB Chair: Professor Kent Rissmiller, Tel. 508-831-5019, Email: kjr@wpi.edu
Human Protection Administrator: Gabriel Johnson, Tel. 508-831-4989, Email: gjohnson@wpi.edu

Advisor: Rodica Neamtu Tel (508) 831-5000 ext. 6802, Email: rneamtu@wpi.edu

Your participation in this research is voluntary.

Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

----- Date: -----
Study Participant Signature

Study Participant Name (Please print)

----- Date: -----
Signature of Interviewer

Appendix F: Recruitment and Screening Forms

Would you like to participate in our MQP study?

For many people with verbal or cognitive impairments, engaging in conversation can be tiresome and time-consuming, limiting their educational, social, and career opportunities. Livox provides a unique, pictogram-based AAC application to facilitate communication through a simple, yet highly customizable interface that also accommodates a wide range of vision and motor impairments. The primary goal of this project is to reduce the time and effort required to communicate for people with disabilities by incorporating a NLP-based multi-label classifier into Livox’s machine-learning infrastructure. Our classifier listens for and detects questions followed by a list of responses, and presents relevant Livox images for each response to users.

The following interview will take 15 minutes. The goal of this study is to determine the impact of the new speech-to-text feature within the Livox application. To do this, we will ask you to complete tasks with both the current Livox application, and targeting the multi-label classifier. We will measure the time and number of clicks necessary to complete these tasks in each scenario. We will conclude the interview with exploratory questions on the user experience of the application. This interview will be anonymous, and your responses will be recorded accordingly to maintain this anonymity.

How long it will take: 15 minutes

If you have any questions or are interested in participating in our study please contact our MQP team at gr-livoxmqp@wpi.edu.

Livox MQP Team:

Zachary Emil - zgemil@wpi.edu

Andrew Robbertz - alrobbertz@wpi.edu

Richard Valente - rcvalente@wpi.edu

Cole Winsor - cwinsor@wpi.edu

Project Advisor:

Professor Rodica Neamtu - rneamtu@wpi.edu

FULL CONSENT SCRIPT FOR SCREENING

Background

You are being asked to voluntarily answer some questions to see if you might qualify to be enrolled in a research study. We are doing a research study in order to evaluate the effectiveness of a newly developed voice-activated feature, developed for the Livox assistive communication system.

If you agree, we will ask you some questions about your ability to safely and efficiently interact with the Livox application and our voice-activated feature. You may feel some discomfort or embarrassment about answering these personal questions, but please know that you may end the screening at any time.

Confidentiality

We will be keeping your answers on file for our records. We will store electronic files in computer systems with password protection and encryption.

If you agree to answer our screening questions, we will share your answers with the following groups of people:

- People who do the research or help oversee the research.
- People from Federal and state agencies, as required by law. Such agencies may include the U.S.

Department of Health and Human Services, the Food and Drug Administration, the National Institutes of Health, and the Massachusetts Department of Public Health.

- Any people if you give us separate permission allowing us to give them your answers.

We might share your answers where we have removed anything that we think would show your identity. There still may be a chance that someone could figure out that the information is about you. Such sharing includes:

- Publishing results in a medical book or journal.
- Using research data in future studies, done by us or by other scientists.

Subject's Rights

Saying yes to this screening and the sharing of your health information does not mean you have to be in the study. Your participation is completely up to you. You can decide to start answering the questions but then stop at any time. Your decision will not affect your ability to get health care or payment for health care. It will not affect your enrollment in any health plan or benefits you can get. However, if you don't answer all the questions, you may not be able to be in the research study.

If you have any questions, please ask them now or at any time you can contact our research team at gr-livoxmqp@wpi.edu. If you would like more information about your rights as a research subject, you may call 617-358-5372 or email medirb@bu.edu. You will be talking to someone at the Boston Medical Center/Boston University Medical Campus IRB. The IRB is a group that helps monitor research.

Signatures

By signing this permission form, you are indicating that

- you have read this form (or it has been read to you)
- your questions have been answered to your satisfaction
- you voluntarily agree to participate in this screening
- you permit the use and sharing of information that may identify you as described

Printed name of subject

Signature of subject

Date

I have personally explained the research to the above-named subject and answered all questions. I believe that the subject understands what is involved in the study and freely agrees to participate.

Signature of person conducting consent discussion

Date

Screening Questions

Please answer the following questions honestly and to the best of your ability.

1. Do you have a history of eye strain from screen use?
2. Do you suffer from any disabilities? (For example, epilepsy, autism, cerebral palsy ...)
3. Do you believe there is any reason you should not participate in this study for your own health?

Appendix G: User Study Interview Scripts

Interlocutor Interview Script

Hello -----,

My name is -----, *introduce other team members present.*

Thank you for agreeing to speak with us today. We are currently completing our Major Qualifying Project, Towards a More Inclusive World: Enhanced Augmentative and Alternative Communication For People With Disabilities Using Artificial Intelligence and Natural Language Processing. The goal of our project is to develop a voice activated feature within the Livox augmentative and alternative communication application. This feature enables the Livox application to listen for questions in conversation and present relevant responses to users.

The goal of this interview is to determine the accuracy of the new feature under a variety of situations. This interview will be anonymous, and your responses will be coded accordingly to maintain this anonymity.

Check they have filled out consent form (Appendix C.1)

Do you have any questions for us before we begin?

If not, begin interview...

Exploratory Questioning

The first phase of this interview will have you ask questions based off a series of topics. The goal of this phase is to discover if an individual with no knowledge of the system will ask questions in a format which can be recognized by our system. We will provide you with the topics for the question, and you will come up with the question.

There are several restrictions which must be followed to use this feature. It is required that you say the name of the user, in this case “John”, before asking the question. While asking the question speak clearly and do not pause. To help with this, take as much time as needed to create and practice your question.

Do you have any questions for us before we begin?

Topics:

1. Lunch
2. Book preferences
3. Math question

Scripted Questioning

The second phase of this interview will have you ask scripted questions. The goal of this phase is to test the real world accuracy of our system for known question formats. We will provide you with a list of questions and have you read them aloud to the application.

Again, there are several restrictions which must be followed to use this feature. It is required that you clearly say the name of the user, in this case “John”, before asking the question. While asking the question speak clearly and do not pause. To help with this, take as much time as needed to practice the question.

Do you have any questions for us before we begin?

1. What do you like better cats or dogs?
2. How are you feeling today: happy, sad, or angry?
3. What do you need to do this morning brush your teeth or go to the bathroom?
4. What is your favorite color: red, green, or maybe blue?
5. What time would you like to take a shower? 8, 8:30, or 9?
6. Do you want to go now or later?
7. What do you want for dinner: a hamburger or a peanut-butter and jelly sandwich?

Informed Exploratory Questioning

The third phase of this interview will be identical to the first, where you ask questions based off a series of new topics. The goal of this phase is to test if an individual with basic knowledge of the system will ask questions in a format which can be recognized. To recap, our feature expects a full question, followed by a list of possible responses. We will provide you with a topic and the response options, and you will formulate a question which encompasses these aspects. Again, there are several restrictions which must be followed to use this feature. It is required that you say the name of the user, in this case “John”, before asking the question. While asking the question, speak clearly and do not pause. To help with this, take as much time as needed to create and practice the question.

Do you have any questions for us before we begin?

1. Ask John his opinion for lunch. You can make hot dogs, hamburgers, or pizza.
2. Ask John his opinion on which book he wants to read. He can read a mystery or adventure novel
3. As John, what the solution to $7+8$ is. Give him the options of 14, 15, and 16

User Interview Script

Hello -----,

My name is -----, *introduce other team members present.*

Thank you for agreeing to speak with us today. We are currently completing our Major Qualifying Project, Towards a More Inclusive World: Enhanced Augmentative and Alternative Communication For People With Disabilities Using Artificial Intelligence and Natural Language Processing. The goal of our project is to develop a voice activated feature within the Livox augmentative and alternative communication application. This feature enables the Livox application to listen for questions in conversation and present relevant responses to users.

Livox is an Augmentative and Alternative Communication application designed to assist people who have difficulty with verbal communication. Livox has text-to-speech that is activated by selecting a tile on the screen. Selecting a tile causes the application to read the tile’s text out loud. Livox is organized in a nested-folder structure, meaning that some tiles will move to a different screen when selected, often to continue or specify a thought. If a desired tile is not on the main screen, you will need to navigate to the appropriate topic to find it.

The goal of this interview is to determine the impact of the new Question Detection feature for the user. To do this, we will ask you to complete tasks with both the Livox application, and the Question Detection feature. We will be measuring the time and effort, number of clicks, necessary to use each. We will conclude the interview with exploratory question on the user experience of the application. This interview will be anonymous, and your responses will be coded accordingly to maintain this anonymity.

Check they have filled out consent form (Appendix C.2)

Do you have any questions for us before we begin?

If not, begin interview...

Livox Application Testing

The first phase of this interview will ask you to complete tasks within the base Livox application. The goal of this phase is to evaluate the time and effort required to use the Livox application without our feature. For this phase we will ask you several questions. Once the question has been completely asked, you will navigate the application to answer with one of the options we gave you.

You now have time to acclimate to the Livox application.

Explain how to use the Livox Application

Do you have any questions for us before we begin?

1. What would you like to have for lunch? Hamburgers or hot dogs?
2. Where do you want to go: School or the Library?
3. Who is your favorite sibling: your Brother or Sister?
4. When do you want to get a haircut? Today or Tomorrow?
5. What do you want to do after school: Play videogames or paint?

Question Detection Feature Testing

The second phase of this interview will ask you to complete tasks within the Question Detection feature. The goal of this phase is to gather comparative data on time and effort when using this feature. For this phase we will ask you several questions. Once the question has been completely asked, you will click one of the presented response options.

Give explanation of how to use the question detection feature: Once we have properly asked a question, you will be taken to a new page after about a second, where the only tiles are those contained in the question. Selecting one of these tiles or pressing the back button will read the tile out loud and return you to the previous screen. The question must be asked again for the screen to reappear.

Do you have any questions for us before we begin?

You now have time to acclimate to the Use of the Question Detection feature.

1. What would you like to have for lunch? Hamburgers or hot dogs?
2. Where do you want to go: School or the Library?
3. Who is your favorite sibling: your Brother or Sister?
4. When do you want to get a haircut? Today or Tomorrow?
5. What do you want to do after school: Play videogames or paint?

Exploratory Questions

The final phase of this interview will ask you questions in order to get your opinion on the user experience of the Question Detection feature.

1. Did you prefer navigating the application or using our Question Detection feature?

2. What part(s) of the Question Detection feature did you find frustrating, if any?
3. What part(s) of the Question Detection feature did you like, if any?
4. Do you have any further comments or recommendations for the Livox application or the Question Detection feature?

Appendix H: Mechanical Turk Questionnaires

Batch 1

This survey focuses on collecting examples of "list questions".

Examples of list questions are:

- "what would you like for dinner pizza, pasta or a hot dog"
- "how would you like your eggs cooked, scrambled, fried, or poached"
- "do you want the cat, golden retriever or hamster"

You are tasked with creating 8 examples of list questions. With the following requirements.

- Use a variety of topics including different categories of objects, places, people and things.
- Don't use the same question word such as "what" exclusively, be creative and use a variety.
- Questions must have a list of things.
- Please attempt to use "compound nouns" in some (not all) questions such as "hot dog" or "chocolate chip cookie"
- Don't worry about punctuation or capitalization
- Lists of objects do not need to be 3 items they can be 2 or more

Example Answer Q1:

1a) List Question:

"what would you like for dinner pizza, pasta or a hot dog"

1b) List Objects:

"pizza, pasta, hot dog"

Batch 2

This survey focuses on collecting examples of "list questions".

Examples of list questions are:

- "are you feeling happy, sad or angry"
- "should I pick you up at eleven or twelve"
- "is your favorite restaurant, McDonalds, Subway, Burger King or Olive Garden"

You are tasked with creating 8 examples of list questions. With the following requirements.

- Use a variety of topics including (not limited to) different categories of numbers, objects, places, people and things.

- Don't use the same question word such as "are" exclusively, be creative and use a variety.
- Questions must have a list of things.
- Please use "compound nouns" in some (not all) questions such as "hot dog" or "chocolate chip cookie"
- Lists of objects do not need to be 3 items they can be 2 or more (use a variety)

Example Answer Q1:

1a) List Question:

"are you feeling happy, sad or angry"

1b) List Objects:

"happy, sad, angry"

1c)

Description

Batch 3

This survey focuses on collecting examples of "list questions".

Examples of list questions are:

- "are you feeling happy, sad or angry"
- "should I pick you up at eleven or twelve"
- "is your favorite restaurant, McDonalds, Subway, Burger King or Olive Garden"

You are tasked with creating 8 examples of list questions. With the following requirements.

- Use a variety of topics including (not limited to) different categories of numbers, objects, places, people and things.
- Don't use the same question word such as "are" exclusively, be creative and use a variety.
- Questions must have a list of things.
- Please use "compound nouns" in some (not all) questions such as "hot dog" or "chocolate chip cookie"
- Lists of objects do not need to be 3 items they can be 2 or more (use a variety)

Example Answer Q1:

1a) List Question:

"are you feeling happy, sad or angry"

1b) List Objects:

"happy, sad, angry"

1c)

Description

IMPORTANT NOTE:

Do not use question words from the examples in your answer, credit will not be given. Specifically do not start sentences with: "Are", "Should" or "Is"

Batch 4

This survey focuses on collecting examples of "list questions".

Examples of list questions are:

- "what would you like for lunch, pizza, a hot dog or hamburgers"
- "how many coins do you need four or five"
- "do you want to go to, McDonalds, Subway, Burger King or Olive Garden"

You are tasked with creating 8 examples of list questions. With the following requirements.

- Use a variety of topics including (not limited to) different categories of numbers, objects, places, people and things.
- Don't use the same question word such as "are" exclusively, be creative and use a variety.
- Questions must have a list of things.
- Please use "compound nouns" in some (not all) questions such as "hot dog" or "chocolate chip cookie"
- Lists of objects do not need to be 3 items they can be 2 or more (use a variety)

Example Answer Q1:

1a) List Question:

"what would you like for lunch, pizza, a hot dog or hamburgers"

1b) List Objects:

"pizza, hot dog, hamburgers"

1c)

Object/Entity

IMPORTANT NOTE:

Do not use question words from the examples in your answer, credit will not be given. Specifically do not start sentences with: "What", "How" or "Do"