Spring 5-18-2012

A Software Framework for Augmentative and Alternative Communication

Adam Loup
University of New Orleans, adloup@uno.edu

A Software Framework for Augmentative and Alternative Communication

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
In
Computer Science
Database Systems and Distributed Applications

By

Adam Loup

B.S. University of New Orleans, 2006

May 2012

Table of Contents

List of Figures

Abstract

By combining context awareness and analytical based relevance computing software, the proposed Augmentative and Alternative Communication (AAC) framework aims provide a foundation to create communication systems to dramatically increase the words available to AAC users. The framework will allow the lexicon available to the user to be dynamically updated by varying sources and to promote words based on contextual relevance. This level of customization enables the development of highly customizable AAC devices that evolve with use to become more personal while also broadening the expressiveness of the user. In order to maximize the efficient creation of conversation for AAC users, the framework provides a lexicon with the ability to obtain words from multiple sources which are then organized according to relevance in a situational context.

Keywords: context awareness, Alternative and Augmentative Communications, AAC device, framework, Android

1 Introduction

Conversation is one of the most important human interactions. For some, this fundamental means of communication becomes impaired due to illness, age, disability or injury. The sciences of Speech and Language Pathology exist to therapeutically treat speech-related disabilities. Alternative and Augmentative Communications (AAC) devices were created to aid both speech pathologists and their patients. The most important function of AAC devices is to allow the creation of conversation for users. These conversations aid in fulfilling the agendas of communication: expressing wants and needs, the transfer of information, social intimacy and social etiquette [9]. The methods of conversation creation are not always straight forward, ranging from predictable, to novel, to intimate. In many cases, the meaning of conversation dramatically changes based on contextual situations as well. Traditionally, AAC devices construct conversation with lexical items that are made available through the device's lexicon. The lexicons are made up of a predefined core vocabulary which has been developed over time by Speech Pathologists to contain the most commonly used words in general conversations. While this core vocabulary helps a great deal with device-aided communication, it is often lacking words that are specific to a user or a conversation context. In some cases the core vocabulary has been found to contain only 33% of the words for a desired communication [1]. Lexicons can be updated; however, the user or the family of the user is often unable to do so. In principle, the lexicon of a device can be enhanced by the addition of new words from updates or even customized by speech pathologists with specialized software. Even with regular updates the lexicon can still return words that are not useful in a situational context.

The goal of this research project is to provide a framework for a context aware system for the devices that many rely on for basic communication. Context awareness enables

applications to transparently adapt results to fit the context of the request [14,15]. The lexicon of the proposed system filters and sorts words based on situational contexts, consisting of the location and the categories the words are used in. The objective is to lessen the burden of device-aided communication by proactively delivering situational dependent words derived from usage history.

2 Background

2.1 Augmentative and Alternative Communication

There are many people, adult and children alike, that have extreme difficulty communicating via oral speech. These speech impediments stem from various congenital and degenerative causes such as; Cerebral Palsy, Stroke, ALS, Autism and Muscular Dystrophy. The American Speech and Hearing Association's (ASHA) defines AAC systems as "an integrated group of components, including the symbols, aids, strategies and techniques used by individuals to enhance communication."[10] For those who are unable to express themselves through speech, AAC devices are an assistive technology to aid and enhance communication capabilities. In some cases AAC devices are used to develop a stronger understanding of communication.

There are two main types of Augmentative and Alternative Communication, unaided and aided. Unaided AAC refers to nonverbal natural communication, such as sign language. Typically, these forms of AAC are used by adults or children that have used of their hands and the motor-skills necessary to perform the gestures used for communication.

Conversely, aided AAC refers to approaches that rely on additional peripherals that render representations of what the user wants to convey. Aided devices also include digital devices that playback recorded or synthetically created speech. To date, most effective means of

language representation in aided AAC devices has been accomplished by presenting the user with alphabet-based symbols. Access to individual words, through spontaneous novel utterance generation (SNUG), has been proven to increase participation in casual conversation and to promote natural language development [10]. Pre-stored messages or phrases rarely meet the needs of conversing in the natural environment and often fail to give the user adequate conversational ability [1].

While effective, whole word symbols require a large symbol set. The choice of vocabulary available to the user is a critical aspect to the success of AAC usefulness. There are two main divisions of vocabulary, core and extended [9,10]. The core vocabulary are the few hundred words that speech pathology research has deemed critical to create general conversation and the majority of social interactions. Extended vocabularies are those words which are used to describe specific items and are used infrequently. Together, these two categories provide a solid foundation for improvements to the augmentative and alternative communication. In addition to these vocabularies, AAC devices must include the full communication ability of the individual.

2.2 Android

Over the last few years, innovations in mobile technology have already greatly aided and enhanced methods of human communication. Modern mobile technology provides an always on and always connected ecosystem. One such class of mobile devices that have gained huge amounts of popularity are tablets. Tablets are mobile personal computers with flat touch screens larger than five inches. Like personal computers, tablets allow users to access the Internet anywhere over wireless networks as well as local storage to store and access data. Many tablets

also come with extra features, such as a gyroscope, GPS, and an accelerometer, that developers harness to provide innovative uses through Apps.

To unify the mobile device market, the Open Handset Alliance and Google worked to design and implement an "open, complete and free platform created specifically for mobile devices." The result of this alliance is the Android Operating System. Android, an open-sourced software architecture platform consisting of Linux-based operating system, middleware and applications designed for mobile devices to create the best possible experience to mobile end users [12]. The Android team has created an open ecosystem for developers to create applications to distribute to any Android-compatible device [11]. The platform provides an interface with the host device's hardware components, such as GPS, as well as access to Google authored and third party authored libraries to increase functionality of developer created applications. Most importantly, Android devices are delivered with text to speech, data storage, HTTP communication and location libraries. The application layer of Android is handled by the Dalvik Virtual Machine optimized for memory and CPU constrained devices. Dalvik provides a full featured Java programming environment and is capable of executing modern Java code. It is important to note that Dalvik is not a J2ME environment and supports most current Java libraries. [11]

In addition to the ability to run standard Java libraries, the two most recent versions of Android, Honeycomb (3.0) and Ice Cream Sandwich (4.0), provide enhancements beneficial to tablet devices with larger and higher resolution screens.

2.3 Relevance Index

Often, obtaining results that match a specific pattern is enough to satisfy a search request. However, In the field of Information retrieval where large data sets are queried, the relevancy of a particular item is more significant than if an item matches a query or not [7]. The Term Frequency times Inverse Document Frequency (TF-IDF) weighting formula is a means of computing the relevance of an item based on terms of the query[7]. The TF-IDF assigns a higher weight to term that occurs more often in a sub set of the larger data set and a lower weight to terms that occur often across the entire data set. First released in 1972, TF-IDF has proven to be very robust and is used in the core of most ranking based search engines [8]. Adding a weighted relevance index to results provides an extra layer of confidence that the results not only satisfy a query but also contain relevance to context in which it as made.

2.4 Stemming algorithm and Lucene Snowball API

When performing Natural Language Processing (NLP) the base form of a word is more important than then conjugated forms of a word. Often the input for NLP contains conjugations of words and the base must be derived via Stemming algorithms. The Porter Stemmer is a step based approach that works on the idea that most English words are comprised of a combination of suffixes [6]. Each step of the algorithm applies rules, defined using the Snowball language, to match and remove a suffix from a word until a stem can be produced [6]. Lucene is a Java based indexing and search tool that performs stemming using the Porter approach and has made the implementation available via the Lucene Snowball API [16].

2.5 Context-Aware Computing

Through the rise in ubiquitous computing the a focus on the physical environment in which an application runs has surfaced. Context-awareness allows applications to adapt to and make assumptions based on situational changes [14, 15]. Context generally consists of a geographic location along with other domain specific information [17]. Applications can respond to context changes in different manners from the availability of certain aspect of an application or the items it manages to events that are triggered when entering or exiting a context [17]. Taking into account the user context allows application to continue to innovate and personalize the use of computers in daily life [15].

2.6 Related Work

The framework proposed in [15] is made up modules, each with a specific task. Context data that originated from sensory data is conditioned and introduced to the system via the Context Normalization Module. The module is also responsible for abstraction or conversion of raw data into recognized types. From this point the Context Aggregation Module combines related contexts by organizing the contexts based on UserId, DeviceId or LocationId, or by domain specific rules. Resulting contexts are added to the Current Context and also to the Filter Module. The Filter Module filters out essential contexts which are then are stored in a history database for future processing. Contexts are not defined purely by sensory data. The Context Inference module uses a rules based approach backed by an established knowledge base to infer contexts, such as the kind of activity the user is doing, based on sensory driven contexts. The patterns of a particular user also play a role in context awareness. The Context Learning Module applies artificial intelligence techniques in order to predict the need for certain services.

Finally, the presence of a particular context can cause actions to occur. These are invoked by the Event Triggering Module.

The context-awareness of the AAC framework is concerned with acting on a particular context more so than building a complex context model. There is a finite number of sensory data from which these contexts are established. Situational contexts are straight forward pairings of a word with the geospatial coordinates a word is used in and the categories traversed by the user in order to get to that word. The driving force behind the AAC framework is the current state of the LexicalContext and how to act upon it to best serve the user. The Lexicon is a connecting point between the WordProviders, the requester of data, and the history of user interaction created by the PhraseAnalyzer.

2.7 Survey of Contextual-Awareness

Context Awareness is used across many disciplines in order to achieve enhancements in life quality, to aid daily activities and to provide a better understanding of current tasks. Context awareness is used in [19] order to maintain a distributed electronic patient record across multiple ubiquitous devices both mobile and stationed at key locations throughout a large hospital. The devices were used make patient records available at any time, in any location.

The Sense Everything, Control Everything (SECE) platform is used to trigger events based on user defined rules [19]. SECE connected previously isolated services in to a useful, user personalized composite service. In one example the user's current location being so far away from a meeting location triggered alerts to be sent to workmates informing them that the user would be late to a meeting.

As seen in [18], context-awareness is used to improve the focus and efficiency of a small meeting by analyzing single person and group interactions though audio and video sensors. Contextual clues were used in order to direct microphones to the current presenter the cameras to a particular object. These automated adjust can perform transparent of user interaction and without disruption of the meeting.

3 Requirements

A one size fits all solution is not practical since the physical ability and literacy level varies from user to user. On the other hand, the underlying operation and lexicon storage is similar across different AAC devices. Having a foundation promotes efficient creation of new applications and improvements to the quality and robustness of existing applications. An established base will allow developers to focus on the special case they are designing for without having to recreate the infrastructure on which every device or application runs.

3.1 Basic Requirements of the AAC Framework

The framework was designed considering three issues critical to the usefulness of applications requiring it. Most importantly, the framework must provide the essential features of AAC devices to impart effective interactive communication. Users should at least be able to access words in the same manner as current AAC devices. Words are divided into logical categories. Categories are collections of words organized based on predetermined logical groupings in contextual situations. For example, words such as “pencil”, “notebook” and “teacher” might be found in the “classroom” and in the “school” categories. The relationships are assembled by speech pathologists and are common among AAC devices. The framework

must provide a means to store and return words based on categories, whenever requested by the user. Words for AAC devices are not solely determined by speech pathologists. There must be a means for words imperative to daily life to be added by the user or a family member. Second, to encourage quick conversation creation the framework must respond quickly to context changes when retrieving and combining results from multiple sources. Therefore, the framework must perform tasks in a non-blocking manner. The user should never have to wait more than 0.1 seconds for words to be available [13]. Finally, a single method of computing relevancy is not enough to handle the needs of all users. Relevancy is an index relating a word to a situational context. The framework must be easily extensible and facilitate other forms of relevance computation [14].

3.2 Problem with the Basic AAC

Current AAC devices provide a closed and rigid lexicon. A static vocabulary does not always adhere to the individual needs of users especially in a variety of social situations. Many users do not always choose words that are present in core vocabularies. Typically, family members and/or speech pathologists are responsible for maintaining the device. They should keep it up-to-date and they should prepare the device for the needs of the user for the next day [9]. While the user can have a direct impact on the words present on a device, there still exists a strong need for fine-grained personalization. Awareness of how the user combines words to create conversation allows AAC devices to fulfill the main purposes of communicative interactions to a higher degree. [5, 9]

In addition to a predefined vocabulary, the size of the screen is also a factor limiting the number of words that are available during conversation construction. The number of words that

can appear on the screen at one time is very limited. Many devices employ dynamic layouts in order to display more meaningful words on the screen. Words in a layout are organized in a specific fashion decided upon by the device designers. The methods of organizing words for layouts vary greatly. Some devices display words divided into categories of semantic frames that the user chooses for the desired conversations [9]. There have also been efforts to improve the selection of words by taking into account details and daily patterns of the user to display words that are more relevant to a situation.

3.3 Context-Aware AAC

Users of AAC devices are often bound to a limited number of conversational contexts. Meanings expressed through conversations are highly dependent upon the context in which it is created. A better understanding of the conversational context gives a greater insight into the intentions and motivations of the speaker. Imparting context awareness and usage data collection into the lexicon can improve word selection with items that are not only relevant to the context but also to how the user has communicated in the past [15]. With continual usage the context-aware AAC can harvest a model of the user based on daily usage patterns in order to bring attention to words relevant to a conversation. Therefore the words that are used most often in a particular location will be marked in order to draw the attention of the user.

4 Design

Drawing upon the different techniques of managing and providing lexical items to AAC users, this framework is designed to be flexible. The framework provides methods to easily add new words into the lexicon. The framework allows the AAC developers to customize the

methods in which word relevance is computed. The framework is intended to be a foundation to create applications on mobile devices and is designed to be lightweight with a low resource footprint.

4.1 AAC Framework Architecture

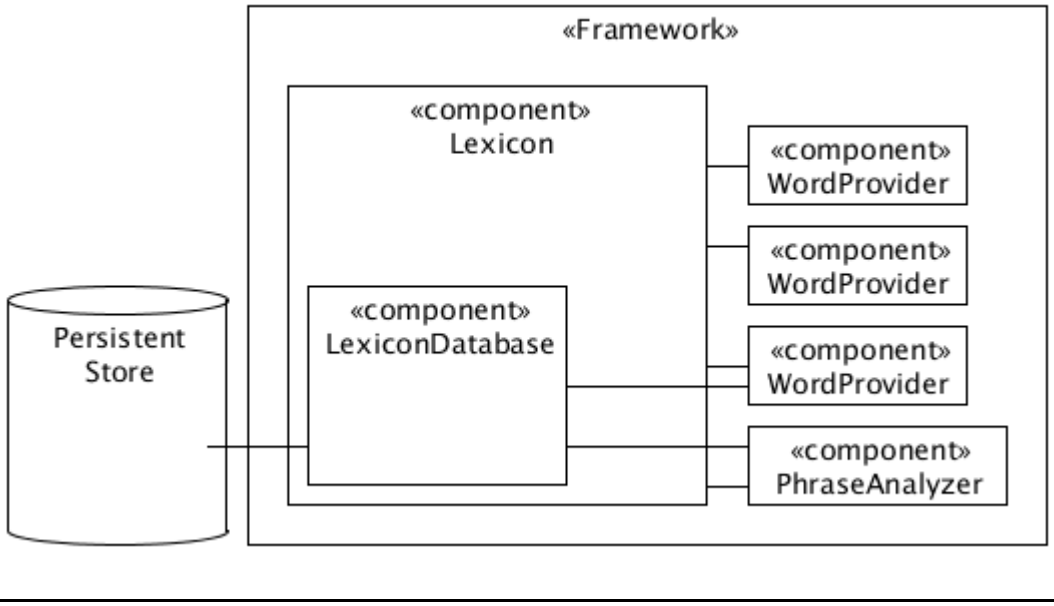

Figure 1. The components of the AAC Framework

Similar to the framework proposed by Park and Lee [15], the AAC framework is compartmentalized into modules responsible for a specific duty. The *Lexicon* is a Context Service that leverages these modules to manage collecting, organizing and maintaining words available to the user by observing how and where words are used [14]. Modules communicate by responding to changes in the context and by events triggered by other components. These modules are also thread safe and work concurrently in order to carry out tasks in a non-blocking fashion. Context awareness is achieved by leveraging multiple contexts into a singular situational context, the *LexicalContext*. Simple context data, in particularly geographic location,

is combined with inferred context information consisting of the actual words spoken and the categories currently used [15]. It should be noted that the Lexical Context is not limited to location awareness. Any number of contextual parameters can be used to define lexical contexts. Contextual information is then passed to the *PhraseAnalyzer*, the filtering and learning module, which records the history of contexts that is later used to enhance adaptation to the user [14, 15]. The Lexicon obtains words by querying *WordProviders*, passing the current location and category. Results are combined and filtered based on their accompanying relevance score and ultimately are presented to the user.

4.2 Lexicon Management

The structure of the data set lends itself to produce a well-defined user profile to apply context awareness. In addition to categories, words are also grouped based on the results of word usage. The Lexicon is tasked with returning a word set relevant to the current situational context. Relevance refers to how often a word is used in a context based on the statistics of past interactions. After a phrase is created, the Lexicon passes it to the *PhraseAnalyser*. At this point the phrase is broken down into the individual words that make up the phrase. First, any stop words are removed. Stop words are functional words used to construct and join concepts but are irrelevant to a situational context [6]. Generally stop words consists of articles, pronouns, interjections, and any other group of function words that are ambiguous in meaning and serve to create grammatical structures. Initially the set of stop words is created from a predefined list used throughout applications that perform natural language processing. However, stop words can be unique to the purposes of an application. To promote personalization, words can be added to the stop word list at any time. For example, through usage history statistics, a user finds that a certain word is used in many different contexts. Because this word is used so often the relevancy

score decreases however, the user still wants to have easy access to it. Adding it to the stop word list will cause the word to be present at all times and will prevent any future usage recording.

Next, Word objects are created for any new items in the word set. New words will not always be base words. Identifying the base of a word can be accomplished with stemming algorithms. The *PhraseAnalyser* handles stemming using the Lucene Snowball API, an implementation of the Porter-Stemming algorithm. The base may or may not exist in the database. In the event that the base word does not exist in the Lexicon, a Word object for the base is also created. Any new Word objects created by the *PhraseAnalyser* are store to the local database, through the persistence layer, for future use. Finally, a usage record is created for each word and category pair remaining, along with the current geographic location. Since one word can have multiple conjugations the base ID of a word is used in place of the word ID. If a usage record for this word/category/location combination already exists, the frequency is incremented. Word usage is kept to track how the user relates words to supplement the predefined grouping, enabling the lexicon to reflect the user on a personal level over time.

Efficient conversation construction greatly depends on the words readily available to the user. AAC devices employ many techniques to improve the rate at which conversation is created. One such method is to display words in an order based on some method of relevance. While helpful, relevance dictated by outside sources is not always beneficial to the personalization of the Lexicon [1, 4]. All words presented to the user for conversation creation are made available through the Lexicon. Word Providers are used to populate the lexicon with words and to facilitate evolving and future methods of discovering meaningful lexical items, which is described in the next section.

4.3 Word Providers

Word Providers are a component of the lexicon that delivers words from various internal or external sources. They dictate which words are contained by the lexicon at a given time. To impart a particular influence on which words are presented, the framework supplies an API to support the creation and management of custom word providers. Desired providers must be registered to the lexicon through the *addProvider* and *removeProvider* methods. Custom word providers must extend the *WordProvider* class and implement the *get* method to return a set of *WordProviderResults*. Each *WordProviderResult* object contains the *Word* object and a corresponding relevance score representing a weighted importance that is calculated at the discretion of the provider. Registered providers are invoked by the lexicon in response to a change in the lexical context (i.e. when the user chooses a different category), and the context is passed to all registered word providers. All communication between the lexicon and word providers is synchronous to prevent the word gathering process from prolonging the construct of conversation due to execution blocking due to the creation of large result sets.

A single change in context can trigger multiple updates to the lexicon. The lexicon combines and sorts the *WordProviderResults* for display to the user as each word provider returns results. When the Lexicon is notified of a change by a WordProvider, the set of words contained by that provider is re-evaluated against the set of *WordProviderResults* contained by the Lexicon. Any new *WordProviderResults* from the originating word provider are added to the lexicon. In the event that a *WordProviderResult* from the provider already exists in the lexicon the relevance scores are compared; the *WordProviderResult* with the lower score is removed while the more relevant result is added to the Lexicon.

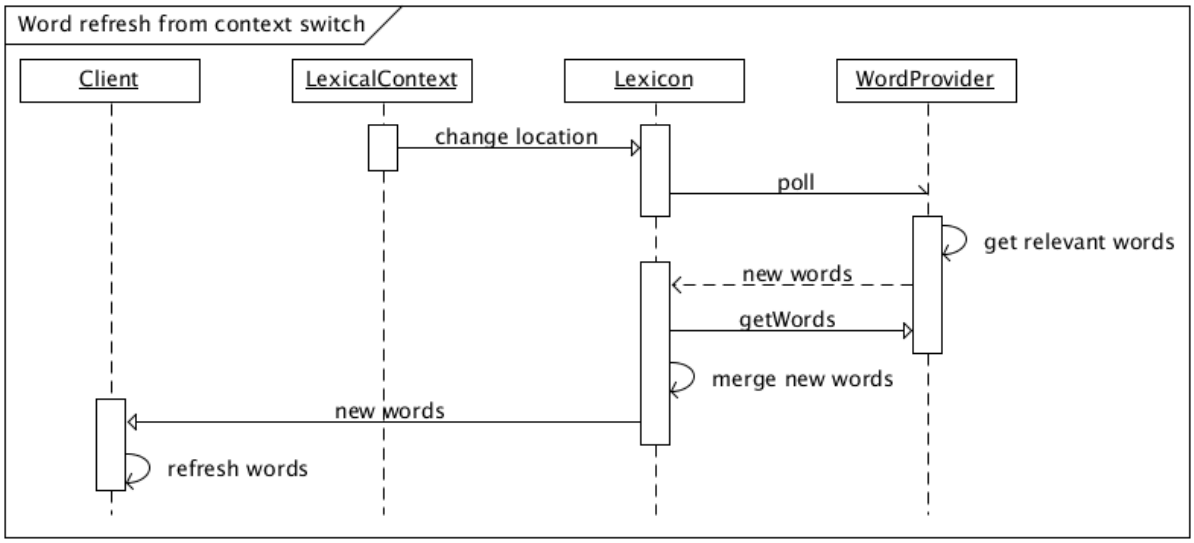

Figure 2. Sequence Diagram showing the interaction between the Lexicon and WordProviders triggered by events from the client user interface.

Word Providers are not only a means to introduce new words into the Lexicon, but also act to create new relationships and improve existing relationships between words. In order to add words from a provider into the Lexicon, the word has to be used in conversation. When the *PhraseAnalyser* is invoked for usage information gathering, it also checks if the word being processed is currently a part of the Lexicon. In the event a new word is found, it is added to the Lexicon with no category associations.

Real-time updating and expansion of the vocabulary along with expandable methods of word discovery aid in lifting the constraints of a limited word pool in conversation. The user is not limited to a predefined lexicon. The combination of these capabilities in a single core library allow the development of a robust AAC devices that encapsulates existing, desired and any future needs of the device end users with the possibility of running on multiple device platforms.

4.4 Reference Implementation of Word Providers

Two word providers are included with the framework, the *LocationWordProvider* and the *LocalWordProvider*. The *LocationWordProvider* returns relevant word results based on the current geographical location of the user. This is achieved through the usage history data collected by the *PhraseAnalyser*. The usage history records the situational contexts in which words were used as well as the frequency in which they were used. The importance of a word to a particular geographic location is determined by using term frequency normalization. This relevance is computed by term frequency times inverse document frequency (*tf***idf*) (Manning, 2008). For purposes of this provider, term frequency (*tf*) is the normalized frequency in which a word has been used in a situational context and is computed as $tf = \sqrt{(\frac{ct_t}{ct_c})}$ where ct_c is the number of times a word has been used in all contexts and ct_t is the total number of times a word has been used in the current context. Document frequency (*df*) is the number of categories, per geographic location, associated with the word divided by the total number of categories (n). The Inverse Document Frequency (*idf*) is then computed as $\log\left(\frac{n}{df+1}\right)+1$. The *LocationWordProvider* uses the result of *tf***idf* for each term as the relevance score for the *WordProviderResult* of the term. Computing the relevance using IDF ensures that a lower importance is given to words that occur greatly throughout all situational contexts. A higher score is given to words that are used more often in a particular situation. The more a word is used in a specific situation the higher the relevance score becomes. The more a word is used in a specific situation the higher the relevance score becomes.

Words stored locally are also accessed through a word provider. The *LocalWordProvider* returns words that are most relevant based on the predetermined categories stored in the word_category table. These relationships serve as a starting point until the usage

history is capable of supplying valid results independently. The framework provides two basic implementations of the word provider class, the basic *WordProvider* and the cache based *CachedWordProvider*. These two classes are meant to be extended in order to create customized Word Providers. For any provider that extends *WordProvider* the *get* method is called every time the lexicon requests words. Conversely, for any word provider that extends the *CachedProvider* the get method is only called when a new context passed to the provider or if the results of a previous request have expired.

Whenever the *CachedWordProvider* is polled for a given context the result set is cached for a set amount of time. For every new context passed to the provider the *get* method is called. This leads to a side effect of the caching the results sets. Word usage frequencies are not updated for a context until the new results are requested. A word list will not immediately be reordered after usage processing has been performed on a phrase.

5 Implementation and Prototype

Implementation of the framework is done in the Java programming language to ensure that the framework will be able to operate on a large number of relatively easy to acquire devices. Any libraries used by the framework, including the framework itself, are compatible with the Standard Java 6.0 Run-time Environment. The framework was developed with the Android platform in mind but does not use any Android specific functionality.

5.1 Persistence Layer

In order to maintain a constantly evolving vocabulary, a local SQLite database managed though ORMLite 4.33, a lightweight Java object-relationship data access layer, is used to persist the Lexicon.

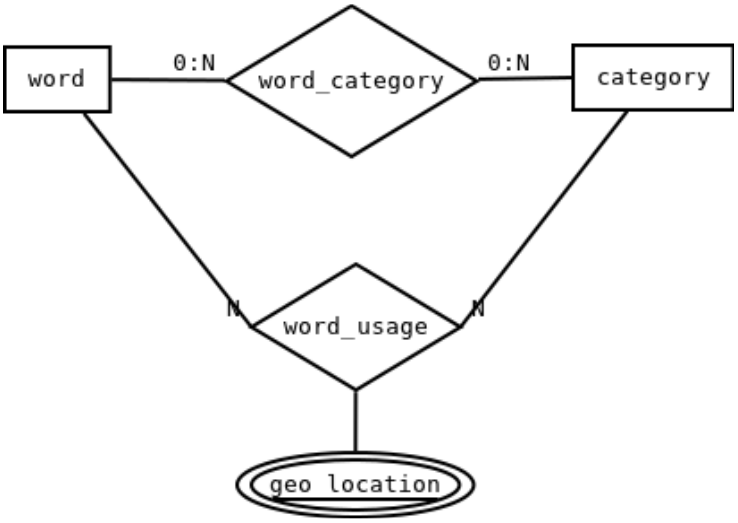

Figure 3. The database structure for Words and Categories

Word objects are the basic elements of the framework which represent single spoken language words. Each Word contains an ID, the text of the word, the word's pronunciation, the locale (language, country, dialect) of the word. Since there is a possibility of words entering the Lexicon from outside sources, the ID is created by the concatenation of the lower case word's text and the word's locale. Similar to Words, Categories have an ID, name and locale. The ID for categories is formed in the same manner as the Word ID, from the locale and the name of the Category. The lexicon maintained by the framework must contain and organize words by category. The many-to-many relationship of Words and Categories is represented by the word_category table. The word_usage table maintains another relationship between Words and

Categories, used to store usage history data, that also includes a geographic location and a frequency.

Access to the database from the components of the framework is handled through the *LexiconDatabase* object. This object is a wrapper for an external persistent data store that contains methods to obtain the Data Access Objects (DAO) for words, categories, word categories and word usages. Each DAO manages the persistence and retrieval of the respective type.

5.2 Prototype

A practical AAC system has been created using the described framework. This prototype was designed to run on a touch screen tablet running the Android operating system. Large screen tablet devices are ideal for the purpose of this prototype. The available screen real estate increases not only the number of words that can be displayed on the screen at one time, but also allows for the inclusion of methods to switch categories faster. The main focus of the interface is to utilize the screen space to its full extent by displaying the most important elements to the user: words and the controls to create speech. The main workspace is divided into two areas; phrase creation and word display. All words are displayed to the user as text which limits the effectiveness of the prototype to literate users with basic understanding of sentence building. However, the underlying framework can be used to support image or icon-based communication construction.

Delivered and customizable Word Providers create a means for multiple sources to contribute relevant words to the conversation. A base lexicon is established by importing libraries that contain the desired categories. The user browses through categories that represent

the conversational contexts. Once a category is selected, the words for the chosen context are displayed. Touching a word causes it to be added to the current phrase, it is vocalized with the speak button. As the user uses an application base on the framework, the context and the words used are recorded in the usage history. The next time the user is in the same context any words that had been used previously will be highlighted in the word list. The brightness of the highlight is based on the computed usage frequency returned by the *WordProviders*, results with a higher frequency are displayed brighter.

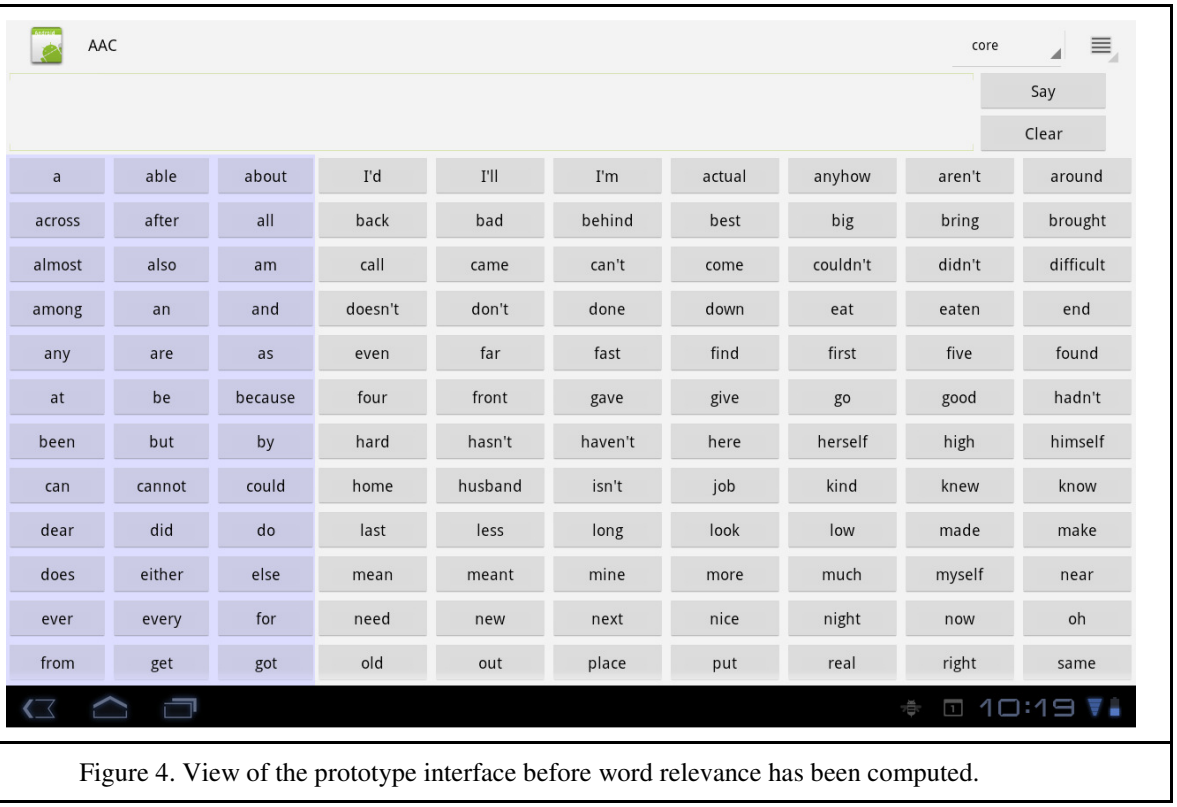

Figure 4. View of the prototype interface before word relevance has been computed.

5.3 Prototype Evaluation

Once the initial core vocabulary has been entered the user is present with a screen displaying all of the stop words on the left side and the words that belong to the "core" category on the right side. The intensity of the background color for a word is dependent on the relevance score, the brighter the color the more the word has been used previously in this context. Various

test phrases were entered. Upon navigating back to the "core" category there is a noticeable difference between words that have and have not been used in the current context.

Figure 5. View of the prototype interface after word usage data has been created. The highlighted words have been computed as relevant to the current *LexicalContext*.

In order to build a useful profile large amount of usage data is needed. Various movie and comic book reviews were collected and parsed to be introduced into the lexicon through the *PhraseAnalyzer*. *LexicalContexts* with predefined categories and geographical locations were created for this test. After the data was imported, the *PhraseAnalyzer* succeeded in both adding new words and creating word usage records. Words were requested from the *LocationWordProvider* using the same context passed to the *PhraseAnalyzer*, the top ten results are shown in Figure 6.

Word	Relevance Score
movie	0.83827
out	0.07824
character	0.07662
way	0.074813
world	0.069434
grey	0.068503
up	0.063997
film	0.062720
Action	0.062194
new	0.062194

Figure 6: Results Found using LexicalContext{locale=en_US, location=GeoLocation{latitude=29.9692345, longitude=-90.1064286}, categories=[core]}.

Filtering the results by relevance scores was able to determine that the word “movie” was most relevance to the user in this contest. Sorting by frequency alone is not enough to calculate meaningful relevance. The *LocationWordProvider* calculates the relevance of a word to a specific context. Based on frequency alone “movie” was the twenty first ranked value. Tthe raw word usage records from the database for the category “core” with the same geographic context passed to the *LocationWordProvider* are shown in Table 1. The frequency for “movie” in this context was 27, compared to the highest frequency word “comic”, with 101. The main factor in determining relevance is how the unique usage a word in a context. The fact that the other twenty words were heavily used in other contexts promoted “movie” to the top for this particular context.

Category Name	Word	Frequency
core	comic	101
core	book	76
core	DC	71
core	new	47
core	World	43
core	way	41
core	up	39
core	app	38
core	story	38
core	more	38
core	time	38
core	out	37
core	art	34
core	title	32
core	make	31
core	really	30
core	first	29
core	before	29
core	year	28
core	good	28
core	movie	27
core	character	27
core	digital	26
core	digitally	26
core	read	26

Table 1

5.4 Creating an Application Using the Framework

Using the framework in an application is very straight forward. A *LexicalContext* singleton instance is obtained to pass to the Lexicon along with a *LexiconDatabase*. The *Lexicon*, as an Observable, follows the Observer pattern alerting the client when the state of the lexicon has changed. All change notifications from the lexicon are intended to trigger UI updates. It is recommended that the client follow the Observer pattern as an Observer, to always have the most recent view of the current lexicon. After the *Lexicon* has been constructed any desired *WordProviders* should be instantiated then added. The code shown in Figure 7 illustrates this process.

```
//create the Lexical Context
LexiconDatabase db = getHelper().getLexiconDatabase();
LexicalContext context = LexicalContext.getInstance(Locale.US);

Lexicon lexicon = new Lexicon(context, db);
lexicon.addObserver(this);

//Add desired WordProviders
LocalWordProvider localProvider = new LocalWordProvider(db);
lexicon.addProvider(localProvider);

LocationWordProvider locationProvider = new LocationWordProvider(db);
lexicon.addProvider(locationProvider);
```

Figure7. Instantiating and configuring the Lexicon to use WordProviders

The Lexicon is designed to react to the events of changes to the *LexicalContext* passed to it during creation. Acting on the LexicalContext by adding new categories or changing the current geographic location, will trigger the Lexicon to make the necessary calls to obtain the words. Words provided by the providers are made available with the *getWords* method of the Lexicon. As mentioned earlier, the framework allows for the construction of customized word providers. To do so the new class must extend the abstract *WordProvider* class and implement

the *getWords* method to return a set of *WordproviderResults*. The code segment in Figure 9 illustrates the corresponding steps.

```
protected Set<WordProviderResult> getWords(GeoLocation location, Category category) {
Set<WordProviderResult> resultSet = new LinkedHashSet<WordProviderResult>();
try {
GenericRawResults<WordProviderResult> usageResults =
this.database.getWordUsageDao().queryRaw(wordResultQuery, resultMapper,
category.getId());

resultSet.addAll(usageResults.getResults());
       usageResults.close();
} catch (SQLException e) {
       ...
}
return resultSet;
}
```

Figure 8. Example implementation of a *WordProvider getWords* method.

.

6 Conclusion

Due to the complexity of natural language, there is no one best method to determine the conversational relevance of words in a situational context. Word Providers allow the Lexicon the flexibility required to achieve a level of personalization that many current Augmentative and Alternative Communication devices do not provide. A centralized mechanism of evaluating relevance based on usage combined with external relevance evaluation and flexible methods of word gathering enable users to expand means of communication in real time. Applying varying methods of relevance computation with a predefined organization of words allows for the discussion of a broader range of topics that may have previously been unavailable to the user. In addition to relevance scoring, usage statistics can be used to generate profiles for word or phrase level prediction to improve the speed at which sentences are formed.

Future work on the framework will include further decoupling of the components that make up the Lexicon and adding means for more customization. Incorporating the extensibility

of the Word Providers to the Phrase Analyzer will allow a finer control on which data is recorded. Allowing developers to customize the data that is stored further enhances the personalization the framework aims to provide. In addition, future work will also introduce ad hoc device discovery capabilities to allow data sharing between users. Data sharing has the potential to add another means of adding new words.

Constantly connected mobile technology presents an opportunity to seamlessly and transparently increase the ability of a user to express themselves through conversation in new and different ways. The user base of AAC devices is comprised a vast variety of unique needs and situations that spans all ages, race and ability. There is no one solution that will work for every user. However, the development of a framework that enhances the usability of the device through content-based relevance over input from multiple word sources streamlines communication creation for multiple types of users. Merging conventional methods of word selection with customizable delivery systems whose results are filtered by content-based relevance systems can provide faster and more accurate access to situational dependent words [1, 2, 3]. By employing dynamic, relevance based methods of word gathering, the framework for augmentative and alternative communications hopes to provide a foundation for devices that tailor themselves to and grow with users to facilitate more personal conversation.

References

[1] Patel, R. & Radhakrishnan, R. (2007):Enhancing Access to Situational Vocabulary by Leveraging Geographic Context. Assistive Technology Outcomes and Benefits, 4(1), 99-114.

[2] Michael J. Pazzani and Daniel Billsus :Content-Based Recommendation Systems

[3] Jun Wang, Arjen P de Vries, Marcel J. T. Reinders :A User-Item Relevance Model for Log-based Collaborative Filtering[10

[4] Linnea R. McAfoose (2004): Using AAC Device Features to Enhance Teenager's Quality of Life. Assistive Technology Outcomes and Benefits 1(1).

[5] Cynthia Overton, Cheryl Volkman, Heidi Silver-Pacuilla, Tracy Gray: Understanding Consumer Needs Through Market Research. Assistive Technology Outcomes and Benefits, 5(1) 2000

[6] Porter, Martin F.: Snowball: A language for stemming algorithms , October (2001) .

[7] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA.

[8] Stephen Robertson: Understanding Inverse Document Frequency:On theoretical arguments for IDF. 2004. Journal of Documentation 60 no. 5, pp 503–520

[9] David R. Beukelman, Pat Mirenda. Augmentative and Alternative Communications: Supporting Children and Adults with Complex Communication Needs, third edition.

[10] Augmentative and Alternative Communication Decisions. http://www.asha.org/public/speech/disorders/CommunicationDecisions.htm

[11] Android, "Home page," Jan. 2010. [Online]. Available: http://www.android.com/

[12] O. H. Alliance, "Home page," Jun. 2010. [Online]. Available: http://www.openhandsetalliance.com/

[13] Card, S. K., Robertson, G. G., and Mackinlay, J. D. (1991). The information visualizer: An information workspace. Proc. ACM CHI'91 Conf. (New Orleans, LA, 28 April-2 May), 181-188.

[14] H. Lei, D. M. Sow, I. John S. Davis, G. Banavar, and M. R. Ebling. The design and applications of a context service. ACM SIGMOBILE Mobile Computing and Communications Review, 6(4):45–55, 2002

[15] Nikos Kalatzis, Ioanna Roussaki, Nicolas Liampotis, Maria Strimpakou, and Carsten Pils. 2008. User-centric inference based on history of context data in pervasive environments. In

Proceedings of the 3rd international workshop on Services integration in pervasive environments(SIPE '08). ACM, New York, NY, USA, 25-30.

[16] Lucene, “Home page,” Nov. 2011. [Online]. Available: http://lucene.apache.org/

[17] B. Schilit, N. Adams, and R. Want. 1994. Context-Aware Computing Applications. In Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications (WMCSA '94). IEEE Computer Society, Washington, DC, USA, 85-90. DOI=10.1109/WMCSA.1994.16 http://dx.doi.org/10.1109/WMCSA.1994.16

[18] Peng Dai, Guangyou Xu. “Context Aware Computing for Assistive Meeting System” PETRA '08: Proceedings of the 1st international conference on Pervasive Technologies Related to Assistive Environments. July 2008.

[19] Omer Boyaci, Victoria Martinez, Henning Schulzrinee, “Bridging Communications and the Physical World” IEEE Internet Computing March/April 2012. 35-43.

Vita

The author was born and raised in Metairie, Louisiana. He obtained his Bachelor's degree in computer science from The University of New Orleans in 2006. During the following years he was employed as a Programmer with the University Of New Orleans University Computing Center, and later as a Software Engineer at Geocent LLC. He entered graduate school in 2007.