Hierarchical Modeling to Facilitate Personalized Word Prediction for Dialogue

Richard G. Freedman and **Jingyi Guo**
School of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA
{freedman, jingyi}@cs.umass.edu

William H. Turkett, Jr. and **V. Paúl Pauca**
Department of Computer Science
Wake Forest University
Winston-Salem, NC 27106, USA
{turketwh, paucavp}@wfu.edu

Abstract

The advent and ubiquity of mass-market portable computational devices has opened up new opportunities for the development of assistive technologies for disabilities, especially within the domain of augmentative and alternative communications (AAC) devices. Word prediction can facilitate everyday communication on mobile devices by reducing the physical interactions required to produce dialogue with them. To support personalized word prediction, a text prediction system should learn from the user's own data to update the initial learned likelihoods that provide high quality "out of the box" performance. Within this lies an inherent trade-off: a larger corpus of initial training data can yield better default performance, but may also increase the amount of user data required for personalization of the system to be effective.

We investigate a learning approach employing hierarchical modeling of phrases expected to offer sufficient "out of the box" performance relative to other learning approaches, while reducing the amount of initial training data required to facilitate on-line personalization of the text prediction system. The key insight of the proposed approach is the separation of stopwords, which primarily play syntactical roles in phrases, from keywords, which provide context and meaning in the phrase. This allows the abstraction of a phrase from an ordered list of all words to an ordered list of keywords. Thus the proposed hierarchical modeling of phrases employs two layers: keywords and stopwords. A third level abstracting the keywords to a single topic is also considered, combining the power of both topic modeling and trigrams to make predictions within and between layers.

Empirically relaxed versions of the developed models are evaluated on training data composed of a mixture of slightly modified dialogues from the Santa Barbara Corpus of Spoken American English. Performance is measured in terms of the number of user interactions (keystroke or touch screen event) required to complete a phrase. We compare their performance against a system employing no prediction.

1 Introduction

The availability of commodity-priced portable computational devices has opened up new opportunities for assistive communication. While these devices are traditionally used to enable long-distance communication, they also offer new ways to assist people with disabilities. It is estimated that between 2.5 to 4.7 million people in the US have conditions severely affecting natural modes of communication, such as speech, gestures, or writing, and require the use of external *augmentative and alternative communication* (AAC) devices to supplement their communication needs (Beukelman and Mirenda 2006). These conditions may be developmental such as autism, L1 syndrome, and cerebral palsy, or acquired conditions such as brain injury, stroke, or amyotrophic lateral sclerosis (ALS). With their increasing computing power and built-in sensors, portable devices have the capability to reshape the world of AAC for a variety of disabilities.

Until recently, AAC devices have been stand-alone tools varying significantly in cost, power, and sophistication. Lower-end devices (ranging from a few hundred to a couple of thousand dollars) offer limited interfaces while high-end devices (up to tens of thousands of dollars) can provide complex interfaces and a great deal of content. However, such interfaces are not easy to use and can often lead to slower communication rates which can have negative impacts on conversations. Also, for disabilities such as ALS where the person has full cognitive capabilities and limited physical mobility, slow communication can be physically stressful and mentally frustrating since one cannot easily express what he intends to say. In response, a new series of assistive technology applications, designed for tablet and phone devices, are being developed, focusing on improving user interfaces over traditional AAC devices as well as reducing costs by utilizing contemporary mobile device development toolkits and methodologies (Pauca and Guy 2012).

The common framework behind an AAC device involves receiving letters and words as input that produce verbal phrases. A standard keyboard can be challenging to use depending on the disability, and it is common for a list of pictures or words to be the primary source of input. However, scrolling through these lists can be just as challenging and time consuming since the interface is limited in how many items may be displayed at one time. Thus *word prediction*, an intent recognition task in natural language processing (NLP) and generation (NLG), has been used to reduce the number of keystrokes necessary to type the phrases. This facilitates the use of the keyboard to type the desired word since only part of the word needs to be present before the entire word is selected from an adapting list of possible words (Garay-Vitoria and Abascal 2006; Trnka 2008a; 2008b; Trnka and McCoy 2007; 2008; Trnka et al. 2006; 2008; 2009). Such methods may also be used to reduce the number of keystrokes used in general purpose text-based communication applications.

Trnka et al. have developed a word prediction system that primarily uses two statistical methods: *trigrams* and *topic modeling* (Trnka 2008a; Trnka et al. 2006) (discussed further in Section 2). As statistical methods, training data is necessary for learning the likelihoods used in the predictions. Due to the personal nature of an AAC device, Trnka et al. have also discussed incorporating the user's text into the training data so that the system can adapt to his/her manner of speaking and vernacular (Trnka and McCoy 2008). However, this requires a large amount of user data since a large corpus of training data is necessary to develop accurate initial likelihood estimates. Thus introducing a bias towards the user will require adding sufficient user data to the corpus so that its sample size dwarfs the initial unbiased sample size. It can take a long time before enough user data becomes available.

Clearly decreasing the size of the initial corpus will reduce the required amount of user data, but this presents a trade-off for less accurate initial likelihood estimates. If the AAC device is not effective "out of the box," then the user may not be as motivated to use it until enough data is produced. Recent work in *abstraction* has shown that learning with a *hierarchical model* can drastically reduce the amount of training data necessary to learn relations and estimate likelihoods (Tenenbaum et al. 2011). The information learned at higher levels of the hierarchy can influence what should be learned at the lower levels through constraints imposed by the hierarchy. For example, the higher-level topic model used by Trnka et al. groups documents together by similar topic, and then lower-level trigram likelihoods are estimated for each group. The application of abstraction and generalization to study higher-level features that influence lower-level features has been used in a variety of artificial intelligence tasks such as hierarchical planning (Knoblock 1991), portfolio-based methods for SAT solving (Xu et al. 2008), hierarchical hidden Markov models (HHMM) (Bui, Phung, and Venkatesh 2004), and metalevel control of anytime algorithms (Hansen and Zilberstein 2001).

In this paper, we add another level to the hierarchical model implied by Trnka et al. that utilizes *keywords*. For our purposes, a keyword is any word in the English language that is neither an article nor a preposition. This level will be placed between the topic model level and trigram level so that each phrase is abstracted to its ordered list of keywords which is abstracted to a single topic. In order to further reduce keystrokes, the user only types the keywords and our word prediction system will be tasked with determining the keyword and then the missing words that belong between adjacent keywords. Uchimoto et al. have performed text generation from keywords in the Japanese language to synthesize sentences that are both grammatically and semantically appropriate based on training from newspaper articles (Uchimoto, Sekine, and Isahara 2002). For each keyword, they generated phrasal units called *bunsetsu* and then matched likely combinations of *bunsetsu*s using several variations of n-grams and dependency rules imposed by the Japanese language. Although the Japanese language is more structured than the English language, we have the advantage of user confirmation during sentence generation; this serves as evidence in our model by fixing the choice of words.

This intermediate level is more informative since keywords are more closely related to the semantic information of the sentence. Thus the topics in the topic model are more relevant to the keywords than the non-keywords, often referred to as *stopwords* in NLP and NLG and *function words* in linguistics. The purpose of stopwords is to provide syntactic structure to the sentence which does not concern topics. Griffiths et al. present this distinction in their composite model that generates sentence frameworks using stopwords from a hidden Markov model and fills in the semantic blanks with keywords from a topic model (Griffiths et al. 2005). We use the opposite approach where the keywords are generated before the stopwords.

Our contribution through this paper is to show how the utilization of abstraction through a hierarchical model can provide reasonable prediction performance with little training data. By reducing the initial number of training samples, user adaptation may occur quicker (due to less data required from the user) in an AAC application. In Section 2, we provide greater detail about assistive technologies, the text prediction problem, and the statistical NLP and NLG techniques used in our text prediction system. Section 3 follows with an overview of our text prediction system's hierarchical model and explanation regarding how our approach works. We present experimental results in Section 4 and conclude with discussion and future work in Section 5.

2 Background

2.1 Assistive Technologies

Assistive technologies represent a broad range of innovations that have been developed to facilitate common activities for persons with physical or cognitive disabilities. A sub-area of assistive technologies, AAC devices are mechanisms to improve both the generation and understanding of language, either in spoken or written form. In the domain of language generation, a spectrum of language components may be presented to an AAC device user, ranging from letters to symbols and words to whole phrases. AAC devices also consider non-typing and non-speaking input mechanisms, such as eye- or head-tracking and touch interfaces.

A broad range of research into intelligent AAC devices is being undertaken. Recent work includes Kanagarajan's investigation (Kanagarajan 2012) of the initial presentation of different symbol sets for symbol-based systems based on features such as the time of day, user location, and recent symbol presses in an attempt to minimize swipes between pages of symbols. The information exploited, such as time and user location, can be easily gathered from most modern tablet and phone devices. Trnka and McCoy have done significant work exploiting predictions of topic for word prediction and augmented keyboards (Trnka, Yarrington, and McCoy 2006).

2.2 Word Prediction Tasks

In textual communication devices such as AAC devices, SMS, and on-line chat, a user must type the words that he/she would normally speak in face-to-face communication. Since the interfaces require pressing keys to type letters or select words from lists, the rate of words expressed per minute is often slower than the verbal rate. For people with disabilities, the reduction can be dramatic compared to others involved in a conversation. Thus methods for speeding up the rate of word expression on textual communication devices are essential.

Word prediction is an intent recognition task commonly used for such speed up. The goal is to predict the word that the user intends to type as soon as possible. A list of m predictions is provided on the interface alongside the keyboard, and the user may select a word from the list by a single keystroke. When the word is not present in the prediction list, the user may select another letter from the keyboard which will provide more accurate information about the word to predict. Various features including current letters typed, previously chosen words, and grammatical derivations of the current sentence have been employed to improve word prediction. A survey of features, techniques, and interfaces used up to 2006 has been written by Garay-Vitoria and Abascal (Garay-Vitoria and Abascal 2006).

2.3 *n*-Grams

Word ordering plays a key role in sentence formation. Due to grammatical rules and semantic situations, specific word patterns make more sense than others which can be utilized in word prediction. For example, it is more likely that an article and noun follow 'go to' while a verb most likely follows 'to go,' yet 'to go the' usually precedes words such as 'extra' and 'whole' which complete several common English idioms.

n-grams are a statistical representation of this phenomenon that considers the ordering of n consecutive words. For word prediction, we find the most likely word in our dictionary D to follow the $(n-1)$ previously viewed words

$$\underset{w_{(n-1)+k} \in D}{\operatorname{argmax}} \mathrm{P}\left(w_{(n-1)+k} \,\middle|\, w_{(n-2)+k}, w_{(n-3)+k}, \ldots, w_k\right)$$

As seen in the example above, greater values of n allow more accurate predictions. However, the number of parameters that must be learned increases by an order of magnitude for each increment of n since there are $|D|^n$ orderings of n consecutive words. This requires both more training data as well as more memory. Lesher et al. have studied the effectiveness of various n and training data sizes and found that for the English language, trigrams ($n = 3$) is sufficient in its trade-off of performance for amount of necessary training data (Lesher, Moulton, and Higginbotham 1999).

2.4 Topic Modeling

Topic modeling, also known as latent Dirichlet allocation (LDA), is a three-level generative probabilistic model for text documents (Blei, Ng, and Jordan 2003). It aims to discover the hidden thematic structure in text corpora. The main idea of a topic model is to associate each word in the document with a latent topic variable. In this way, each document could be represented as a mixture of K "topics." Topic modeling was first developed as a way to automatically summarize topics in scientific papers (Griffiths and Steyvers 2004). Currently some researchers also apply topic models to areas including social networks and unconventional data such as Twitter and IRC.

The generative process of a topic model is as follows: for corpus $W = \{w_0, w_1, \cdots, w_N\}$, the j^{th} word in the i^{th} document $w_{i,j}$ is generated by latent topic variable $z_{i,j}$ whose value is drawn from a multinomial distribution with parameter $\phi_{z_{i,j}}$. $\phi_{1:K}$ is the topic distribution over all words in the vocabulary and $\phi_{k,v}$ represents the probability of token type v appearing in the k^{th} topic. $z_{i,j}$ is drawn from a multinomial distribution with parameter θ_i. Here $\theta_{1:D}$ denotes the topic distribution associated with each of the D documents and $\theta_{i,t}$ represents the probability of topic t appearing in document i. $\phi_{1:K}$ and $\theta_{1:D}$ are drawn from Dirichlet distributions with hyperparameters β and α respectively.

2.5 Hierarchical Abstraction of Data

Objects can be clustered based on their shared features. These clusters form generalized representations of the objects omitting the specifics that make them distinct. For example, 'jump,' 'run,' and 'play' are all verbs while 'boy,' 'store,' and 'ball' are all nouns. These clusters can often be grouped further into higher-level abstractions; both verbs and nouns are words. This yields a hierarchy of abstraction where the lowest level contains the objects themselves and each level above is a generalization of the level below containing clusters of similar objects/clusters.

Underlying patterns and rules that govern the functionality and design of objects can be viewed at different levels of abstraction in this hierarchy. If we know the highest level of abstraction to which each pattern and rule applies, then we can impose constraints at lower levels of the hierarchy. That is, we can use the smaller abstracted framework to gain insights into relationships between a larger number of lower-level attributes. One of the greatest benefits of these constraints is reduction in computational complexity at the lower-levels of the hierarchy. For example, if we consider the rule that the English language does not allow two verbs to follow consecutively, then we can prune the list of words to consider following 'jump' in a sentence. Rather than computing the likelihood of "... jump w" for every word w in our dictionary, we can prune the set of values w may be assigned to all non-verbs in our dictionary. As most data collected for machine learning applications belong to the lower levels of the hierarchy, these constraints can aid the learning process by finding the "big picture" at higher-level abstractions of the data which will reduce the number of parameters that need to be learned at the lower levels of abstraction.

3 Methods

3.1 Hierarchical Model Derivation

Our hierarchical models of a phrase consist of two to three levels: (topic), keyword, and stopword. We define a phrase

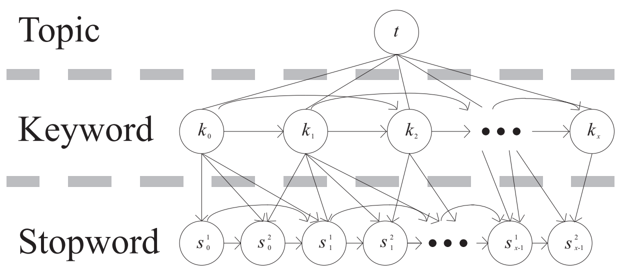

Figure 1: The graphical representation of the hierarchical model of a phrase. The levels of abstraction of a phrase depicted by the model from highest to lowest are topic, keywords, and entire phrase (both stopwords and keywords). Directed edges indicate conditional dependence and undirected edges indicate joint dependence. The two-level model simply removes the topic layer and all undirected edges.

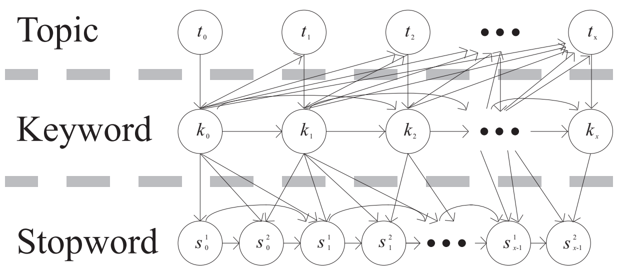

Figure 2: The graphical representation of the relaxed empirical hierarchical model. The joint dependencies between the topic and keyword levels are replaced by conditional dependencies that allow evidence to be observed in real time. This allows incremental predicitions of the phrase topic and current keyword.

to be the ordered collection of all words typed until the user submits the text for transmission; this can range from a single word such as 'yes' to several sentences such as a formal greeting. The graphical representation of our three-level hierarchical model may be seen in Figure 1.

The topic level is the highest level in our hierarchy with a single node t denoting the topic of the phrase being typed into the device. A single phrase in a conversation will usually relate to a particular topic such as greeting, shopping, or cooking. We use unsupervised methods to cluster the training phrases from the corpus into a predetermined number of topics as described in Section 2.4. That is, we determine the topic of a phrase by its keywords.

Inversely, the topic of the phrase will clearly have an impact on the keywords used. For example, fruits are more likely to be mentioned when discussing shopping or cooking rather than sleeping. Trnka used topics in this way to learn a specific trigram table for each topic (Trnka 2008a; Trnka, Yarrington, and McCoy 2006). However, we use topics to learn frequencies of each keyword per topic instead. The primary reason for this relaxation is that we want to use a smaller corpus for training. If we have a separate trigram table for each topic, then there would be $|T| \cdot |D|^3$ parameters to learn rather than $|T| \cdot |K|$ parameters for the separate unigram tables, where T is the set of topics, D is the dictionary of all words, and $K \subset D$ is the set of all keywords.

The secondary reason is that we will learn a single trigram table for all the keywords. At the keyword level of the hierarchy, we have x nodes representing the ordered keywords in the phrase $k_0, k_1, \ldots, k_x$ with a trigram dependency relationship between them. Trigrams over keywords are more informative than trigrams over all words because stopwords have syntactic rather than semantic purposes. Given two consecutive stopwords "to the," there is a wide variety of plausible predictions using trigrams including 'boy,' 'dog,' 'store,' 'left,' etc.. 4- or 5-grams would be more useful for pruning the list of possible predictions in this case with such consecutive word lists as "give it to the" or "let's go to the." On the other hand, removing the stopwords contracts these five-word lists to three-word lists and the semantics of these keywords gives us the same pruning options. The keywords 'boy' and 'dog' are more reasonable for "give it" while 'store' and 'left' are more likely to follow consecutive keywords "let's go."

Utilizing a single trigram table for all keywords also allows us to easily extract a two-level version of the model in which the topic level is removed. The incentive behind this is that LDA contains many parameters to learn for each word and topic. Furthermore, the trigrams between keywords may intrinsically denote the topic locally within the phrase.

The stopword level of the hierarchical models simply predicts which stopwords, if any, should be placed between two consecutive keywords from the keyword level. By restricting our choice of stopwords to just the articles and prepositions of the English language, we are able to assume that there are at most two stopwords between consecutive keywords: a preposition followed by an article (as part of a prepositional phrase). Thus there are two nodes s_i^1 and s_i^2 in the stopword level between nodes k_i and k_{i+1} in the keyword level. Because it will not always be the case that both stopwords will be used, these nodes may also be assigned an empty string token ϵ. Considering how to adjust this assumption for additional closed classes of function words such as conjunctions will be left as future research. The dependency for stopwords are trigrams over all words without conditioning for an observation of ϵ. To enforce this, $s_i^1 = \epsilon$ only if $s_i^2 = \epsilon$ and s_i^1 depends on k_i, s_{i-1}^2, s_{i-1}^1, and k_{i-1} (see the conditional probability tables in Section 3.3).

Empirical Relaxation Due to the joint dependencies between nodes in the topic and keyword layers, prediction of values for these variables would resemble inference over a Markov Random Field. However, this is the *Maximum-a-Posteriori* (MAP) problem which is known to often be computationally intractable. Although there exist many efficient MAP estimation algorithms, we will instead use a relaxed version of the hierarchical model that takes advantage of the user's involvement with predictions. Since the user selects keywords one by one, each selection serves as evidence, an observed (and thus fixed) assignment of a value to a variable.

With this incrementally obtained evidence, we may better predict the topic of the sentence. Thus we denote t_i as the topic of the phrase with respect to observed keywords $k_0, k_1, \ldots, k_{i-1}$. We may then use t_i to make a more accurate prediction of keyword k_i. So instead of assigning one topic per phrase, the relaxed empirical model assigns one topic per keyword. Due to this, we use LDA to implement the topic model in the highest level of the hierarchical model. In terms of modifying the graphical representation in Figure 1, node t is replaced by nodes $t_0, t_1, \ldots, t_x$ and there are directed edges from k_i to t_j for all $i < j$ as well as from t_i to k_i for all $i \in \{0, 1, \ldots, x\}$; there are no undirected edges in the relaxed empirical hierarchical model. The graphical representation is presented in Figure 2. The two-level hierarchical model remains unchanged since it lacks any joint dependencies.

3.2 Corpus for Training

As in any machine learning application, the relevancy of the training data is crucial. The closer the relationship between the training corpus and the user's typical conversational word and phrase usage, the greater the accuracy of the model from the user's viewpoint. We do not have any user input a priori and must instead use a default training corpus, after which user adaptation occurs by training on-line with additional phrases as they are generated by the user. Without decent "out of the box" performance from the initial corpus, the user may not be inclined to use the application to the point that user adaptation can improve it.

However, dialogue spans a broad range of topics and multiple applications that all require different training corpora. For example, Kerr and Szafron use fantasy genre movie scripts for their application of dialogue generation for videogame characters (Kerr and Szafron 2009). In our case, users of AAC devices will most often participate in everyday conversations while using the device. Compilations of such conversations have been recorded and transcribed over the years into a few corpora. Yet Trnka and McCoy (Trnka and McCoy 2007) as well as Vertanen and Kristensson (Vertanen and Kristensson 2011) have noted that AAC dialogue can differ from the contents of these corpora; no formal corpus of actual AAC input currently exists. Vertanen and Kristensson have publicly released a crowd-source generated AAC-themed corpus that empirically improved their text prediction results. Trnka showed that the synergistic effects of combining AAC examples with non-AAC dialogue have also been effective. However, the crowd-source generated AAC-themed corpus was developed for row-column scanning AAC devices rather than those that use a keyboard input. Thus our training corpus is a hybrid of select conversations from the Santa Barbara Corpus of Spoken American English (SBC) (Du Bois et al. 2000; 2003; Du Bois and Englebretson 2004; 2005). We chose the SBC for its variety of discussions and simple linguistic annotations which facilitated setup for training in our hierarchical model. Using fourteen of the dialogues, the corpus currently contains 6477 phrases of varying lengths.

Corpus Setup and Training To train our hierarchical model on the SBC selections, we have to make additional modifications to the text. In particular, the dialogues have to be stratified by speaker, special keyword tokens may need to be inserted for specific situations, and empty string tokens need to be inserted where necessary. The stratification by speaker simply sorts the dialogue so that all phrases spoken by a single individual are listed consecutively. This mimics an AAC device since only its user's input is known; the phrases of others involved in the conversation are not observed. By having a single stream of dialogue for a speaker, relations regarding transition of topic for a user may be learned from the successive phrases presented. Although we currently do not utilize this feature, it may be useful in dynamic variations of our hierarchical models.

At the keyword-level, stopwords are ignored. A complete list of stopwords based on Huddleston and Pullum's analysis of the English language (Huddleston and Pullum 2002) are included in the supplement (*http://people.cs.umass.edu/~freedman/freedmanSupplement.pdf*). The alterations to the text at this level serve sentence parsing and generalization purposes. We use a special keyword token to denote punctuation for terminating sentences since some keywords are more common at the start and/or end of sentences. Similarly, special keyword tokens signifying the start and end of each phrase are inserted into the phrase; due to the trigram model, two special start tokens and one special end token are added. Special keyword tokens generalizing specific names, numbers, locations, and times can trigger the replacement of the standard list of keyword predictions with other features of the AAC device such as contact lists, number pads, maps, etc.. However, the use and recognition of these tokens are left for future research. To learn the trigrams over keywords for training our models from the corpus, we simply count the frequencies of all sequences of three consecutive keyword tokens.

The stopword-level modifications are used to provide the model with values for all its variables. Due to our model's two-stopword assumption, every pair of consecutive keywords needs two stopwords between them. However, it is often the case that there are fewer stopwords. Empty string token ϵ will be inserted in place of the missing stopwords. When there is only one stopword present, ϵ will follow it to maintain consistency. In the few cases where more than two stopwords are found, stopwords are removed by human selection until only two remain. To learn the trigrams over all words for training our models, we count the frequencies of all sequences of three consecutive non-ϵ tokens (that is, empty strings are ignored). In order to find the trigram likelihood that some $s_i^1 = \epsilon$, we count the sequences of three consecutive tokens where only the third token is ϵ (other ϵ are omitted).

3.3 Word Prediction with Hierarchical Model

The word prediction system for our hierarchical model receives keystrokes from the user at the keyword level and has two prediction phases. When a phrase is initialized, we default the values of k_0 and k_1 to the special start token and set $s_0^1 = s_0^2 = \epsilon$ so that the trigram model may be applied for predicting the first keyword of the phrase before the user provides any input. Due to the joint dependence between topic and keywords, we predict the tuples of topics and keywords that maximize the joint likelihood of topic t and current keyword k_c (which are assumed to be independent):

$$\begin{aligned}&\underset{(t,k_c)\in T\times K_{L_s}}{\operatorname{argmax}} \; \mathrm{P}\left(t,k_c\left|k_{c-1},k_{c-2},\ldots,k_0,H,I\right.\right)\\ =\;&\underset{(t,k_c)\in T\times K_{L_s}}{\operatorname{argmax}} \; \mathrm{P}\left(t\left|k_{c-1},k_{c-2},\ldots,k_0,H,I\right.\right)\\ &\qquad\cdot \mathrm{P}\left(k_c\left|k_{c-1},k_{c-2},t\right.\right)\\ =\;&\underset{(t,k_c)\in T\times K_{L_s}}{\operatorname{argmax}} \; \mathrm{P}\left(t\left|k_{c-1},k_{c-2},\ldots,k_0,H,I\right.\right)\\ &\qquad\cdot \mathrm{P}\left(k_c\left|k_{c-1},k_{c-2}\right.\right)^{1}\cdot \mathrm{P}\left(k_c\left|t\right.\right)\end{aligned}$$

where K_{L_s} is the set of keywords that start with the sequence of s typed letters L_s, H is the set of hyperparameters, and I is other information provided by the AAC device (time of day, location, etc.) and/or previously typed phrases (topic history, etc.). Since all keywords k_0 through k_{c-1} and stopwords s_0^1 through s_{c-2}^2 have been selected by the user as part of the phrase, we may consider them as evidence in the model. The m most likely predicted keywords in these tuples are displayed for the user to select. If there are less than m possibilities, then the remaining prediction slots are filled with the most likely predicted keywords in $\left(K_{L_{s-1}}-K_{L_s}\right)$ under the assumption that the previous keystroke was an error. This is repeated for each set of keywords $\left(K_{L_{s-i}}-K_{L_{s-i+1}}\right)$ until all m prediction slots have an option.

In the relaxed empirical hierarchical model, we only find keyword k_c and topic t_c at each step. Thus, since the user does not select the topic with the keyword, it suffices to predict k_c by summing over its joint likelihoods with each topic:

$$\underset{k_c\in K_{L_s}}{\operatorname{argmax}} \sum_{t_c\in T} \mathrm{P}\left(t_c,k_c\left|k_{c-1},k_{c-2},\ldots,k_0,H,I\right.\right)$$

For the keywords that serve as stopwords for LDA (that is, the keyword is a function word that is neither a preposition nor an article - please refer to the supplement for a complete list), only the n-grams are applicable because function words appear frequently enough in documents of any topic. So for these keywords and all $t_c\in T$, $\mathrm{P}\left(t_c\left|k_{c-1},k_{c-2},\ldots,k_0,H,I\right.\right)\cdot \mathrm{P}\left(k_c\left|t_c\right.\right)=1$. This also holds for all keywords when using the two-level model since the topic variables are not present.

Once keyword k_c has been assigned by either selecting it from the prediction list or pressing the space bar, the second prediction phase begins. Using the conditional dependencies between the keyword and stopword levels in tandem with the trigrams of all words, we find the stopwords that maximize their joint likelihood between keywords k_{c-1} and k_c:

$$\begin{aligned}&\underset{s_{c-1}^1,s_{c-1}^2\in S}{\operatorname{argmax}} \; \mathrm{P}\left(s_{c-1}^1,s_{c-1}^2\left|k_c,k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)\\ \propto\;&\underset{s_{c-1}^1,s_{c-1}^2\in S}{\operatorname{argmax}} \; \mathrm{P}\left(k_c\left|s_{c-1}^2,s_{c-1}^1,k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)\\ &\qquad\cdot \mathrm{P}\left(s_{c-1}^1,s_{c-1}^2\left|k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)\\ =\;&\underset{s_{c-1}^1,s_{c-1}^2\in S}{\operatorname{argmax}} \; \mathrm{P}\left(k_c\left|s_{c-1}^2,s_{c-1}^1,k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)\\ &\qquad\cdot \mathrm{P}\left(s_{c-1}^2\left|s_{c-1}^1,k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)\\ &\qquad\cdot \mathrm{P}\left(s_{c-1}^1\left|k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)\end{aligned}$$

where $S=(D-K)\cup\{\epsilon\}$ and the proportionality simplification is a consequence of the conditional variant of Bayes's Rule. Because our trigram tables over all words do not allow conditioning on an observed empty string, we have the following conditional probability tables for the probabilities (X represents "does not matter" and any value assignments not listed imply a probability of 0):

$\mathrm{P}\left(k_c\left|s_{c-1}^2,s_{c-1}^1,k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)$

s_{c-1}^2	s_{c-1}^1	s_{c-2}^2	s_{c-2}^1	Probability
$\neq\epsilon$	$\neq\epsilon$	X	X	$\mathrm{P}\left(k_c\left|s_{c-1}^2,s_{c-1}^1\right.\right)$
ϵ	$\neq\epsilon$	X	X	$\mathrm{P}\left(k_c\left|s_{c-1}^1,k_{c-1}\right.\right)$
ϵ	ϵ	$\neq\epsilon$	$\neq\epsilon$	$\mathrm{P}\left(k_c\left|k_{c-1},s_{c-2}^2\right.\right)$
ϵ	ϵ	ϵ	$\neq\epsilon$	$\mathrm{P}\left(k_c\left|k_{c-1},s_{c-2}^1\right.\right)$
ϵ	ϵ	ϵ	ϵ	$\mathrm{P}\left(k_c\left|k_{c-1},k_{c-2}\right.\right)^1$

$\mathrm{P}\left(s_{c-1}^2\left|s_{c-1}^1,k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)$

s_{c-1}^2	s_{c-1}^1	s_{c-2}^2	s_{c-2}^1	Probability
X	$\neq\epsilon$	X	X	$\mathrm{P}\left(s_{c-1}^2\left|s_{c-1}^1,k_{c-1}\right.\right)$
ϵ	ϵ	X	X	1

$\mathrm{P}\left(s_{c-1}^1\left|k_{c-1},s_{c-2}^2,s_{c-2}^1,k_{c-2}\right.\right)$

s_{c-1}^1	s_{c-2}^2	s_{c-2}^1	Probability
X	$\neq\epsilon$	$\neq\epsilon$	$\mathrm{P}\left(s_{c-1}^1\left|k_{c-1},s_{c-2}^2\right.\right)$
X	ϵ	$\neq\epsilon$	$\mathrm{P}\left(s_{c-1}^1\left|k_{c-1},s_{c-2}^1\right.\right)$
X	ϵ	ϵ	$\mathrm{P}\left(s_{c-1}^1\left|k_{c-1},k_{c-2}\right.\right)$

The m most likely values of $s_{c-1}^1\in S_{L_s}$ that maximize $\sum_{s_{c-1}^2\in S}\mathrm{P}\left(s_{c-1}^1,s_{c-1}^2\left|\ldots\right.\right)$ are listed for the user to select. Then s_{c-1}^1 is used as additional evidence when listing the m most likely values of $s_{c-1}^2\in S_{L_s}$. After predicting the final two stopwords s_{x-1}^1 and s_{x-1}^2 that may preceed the special end token (when the user selects the button to submit or process the phrase), the text prediction process starts over and reinitializes the model for a new phrase.

4 Experiment

4.1 Performance Metric

We measure performance as the *percentage of keystrokes used* which is simply the ratio of the minimum number of keystrokes necessary to generate the sentence with the text prediction system to the minimum number of keystrokes necessary to generate the sentence without a text prediction

[1]The probability $\mathrm{P}\left(k_c\left|k_{c-1},k_{c-2}\right.\right)$ for predicting stopwords is not equal to the one used for predicting keywords. Stopword prediction uses trigram counts of all words in the phrase while keyword prediction uses trigram counts of the keyword abstraction of the phrase. For example, the trigram 'I,' 'ran,' 'it' will not be counted in the phrase "I ran away from it," but it will be counted in the keyword abstraction since 'away from' is removed.

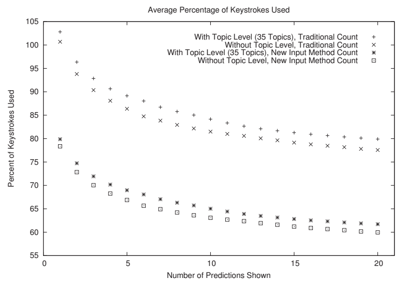

Figure 3: The average percentage of keystrokes used in the experiment. The change in savings indicates that at least 15 predictions should be displayed for best performance.

system. A keystroke is defined as the selection of a keyboard button or predicted word. This measure is commonly used for evaluating the effectiveness of text prediction systems; there is a lower bound to this percentage since at least one keystroke is necessary to select each prediction.

Normally, the minimum number of keystrokes needed to generate a sentence without a text prediction system is simply the number of characters, including white spaces. However, our two-stopword assumption and backtracking to insert stopwords alters this. In particular, after typing all the characters of keyword k_c and a space, the user types the characters of preceeding stopwords s^1_{c-1} and s^2_{c-1}. When one of these is ϵ, a single space is typed to indicate "no more stopword text." This extra space increases the minimum number of keystrokes since many pairs of consecutive keywords have less than two stopwords between them. For example, "hello world!" increases from fourteen keystrokes to nineteen keystrokes; this is detailed in the supplement.

4.2 Procedure and Results

To test the text prediction performance of our hierarchical models, we ran ten trials using our SBC-derived corpus. In each trial, 5182 phrases were randomly selected for training and the remaining 1295 phrases were tested. Both models received the same set of phrases for training and testing. During testing, the word was selected as soon as it was predicted.

Initial results of our current implementation may be seen in Figure 3. Both methods for determining minimum keystrokes are plotted for comparison. Because the new input method requires more keystrokes, the average savings are better for it than the traditional method. However, the number of keystrokes used in the worst case with prediction is, on average, equal to the number of keystrokes traditionally used without prediction. It may also be noted that the hierarchy performed better without the topic level. As discussed in Section 3.1, this could be a consequence of the large amount of data necessary for learning topic models.

5 Conclusion

We have introduced two new models for use in text prediction tasks that utilize a hierarchy of topics, keywords, and stopwords that designate different levels of abstraction of a user's dialogue. The hierarchy requires fewer learning parameters in order to enable efficient initial performance with less training data. A consequence of this is that users, especially those with disabilities who use AAC devices for communication, are able to own devices that can quickly be personalized in order to facilitate their communication experience. The results show reasonable performance with only 5182 training samples, but the extra keystrokes for the new input method prevent reducing the number of necessary keystrokes to a minimum. Now that the initial framework is developed, model modifications and larger-scale testing with corpora of various sizes is possible. Since the SBC is freely available to the public, we will also make our adaptation of the corpus freely available as a possible standard test for text prediction system performance. Such standard tests do not yet exist for evaluating AAC devices.

5.1 Future Work

In future work, we hope to further develop the abstraction hierarchy in order to maximize performance and minimize the size of the training corpus. One way to do this is to insert a *morpheme* and/or *lexeme* level above the keyword level of the hierarchy. Morphemes are atomic definition units of words. Uchimoto et al. (Uchimoto, Sekine, and Isahara 2002) used them as an abstraction of our keywords definition in order to generate *bunsetsu*s. Lexemes are groups of word variants such as representing all conjugations of a verb with its infinitive. This higher-level information serves as latent variables for the observed words giving our models a structure resemblant of HHMM's (Bui, Phung, and Venkatesh 2004). A second way to improve the model would be to allow additional closed classes such as conjunctions to be stopwords and allow the number of stopwords to vary between consecutive keywords.

Besides improving the model, we must also revise the interface to reduce the minimum keystroke increase. In particular, we plan to investigate accurate insertion of predicted ϵ's so that the user is not forced to select them. We also intend to integrate information from the device via special keyword tokens as described in Section 3.2. Lastly, we plan to test our word prediction system with human subjects as Trnka et al. (Trnka et al. 2008) did to evaluate how efficient and intuitive it is for users. It is crucial that typing at the keyword level is as simple as typing the entire phrase sequentially; it may not be the case that most users consciously think from an abstracted point of view. Likewise, we would be interested to see how long it takes for user adaptation to become apparent and its impacts on the human subject's communication rate.

Acknowledgments

The authors thank Shlomo Zilberstein and the anonymous reviewers for their comments that have improved this paper. They additionally thank Duane Szafron and Christopher Kerr for their advice about creating a dialogue corpora as well as Keith Trnka for his insight regarding the reasons for the lack of stopwords in the crowd-source corpus. Support for this work was provided in part by the National Science Foundation under grant IIS-0915071.

References

Beukelman, D. R., and Mirenda, P. 2006. *Augmentative & Alternative Communication: Supporting Children & Adults with Complex Communication Needs*. Paul H. Bookes.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3:993–1022.

Bui, H. H.; Phung, D. Q.; and Venkatesh, S. 2004. Hierarchical hidden markov models with general state hierarchy. In *Proceedings of the 19th National Conference on Artificial Intelligence*, AAAI'04, 324–329. AAAI Press.

Du Bois, J. W., and Englebretson, R. 2004. *Santa Barbara Corpus of Spoken American English, Part 3*. Philadelphia: Linguistic Data Consortium.

Du Bois, J. W., and Englebretson, R. 2005. *Santa Barbara Corpus of Spoken American English, Part 4*. Philadelphia: Linguistic Data Consortium.

Du Bois, J. W.; Chafe, W. L.; Meyer, C.; and Thompson, S. A. 2000. *Santa Barbara Corpus of Spoken American English, Part 1*. Philadelphia: Linguistic Data Consortium.

Du Bois, J. W.; Chafe, W. L.; Meyer, C.; Thompson, S. A.; and Martey, N. 2003. *Santa Barbara Corpus of Spoken American English, Part 2*. Philadelphia: Linguistic Data Consortium.

Garay-Vitoria, N., and Abascal, J. 2006. Text prediction systems: A survey. *Univers. Access Inf. Soc.* 4(3):188–203.

Griffiths, T. L., and Steyvers, M. 2004. Finding scientific topics. *PNAS* 101(suppl. 1):5228–5235.

Griffiths, T. L.; Steyvers, M.; Blei, D. M.; and Tenenbaum, J. B. 2005. Integrating topics and syntax. In Saul, L. K.; Weiss, Y.; and Bottou, L., eds., *Advances in Neural Information and Processing System 17*. Cambridge, MA, USA: MIT Press. 537–544.

Hansen, E. A., and Zilberstein, S. 2001. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence* 126(1-2):139–157.

Huddleston, R., and Pullum, G. K. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press.

Kanagarajan, S. 2012. Assistive mobile interface using machine learning. Master's thesis, Wake Forest University, Winston-Salem, NC 27109.

Kerr, C., and Szafron, D. 2009. Supporting dialogue generation for story-based games. In *Proceedings of the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference*, AIIDE-09, 154–160.

Knoblock, C. A. 1991. Search reduction in hierarchical planning solving. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 686–691.

Lesher, G. W.; Moulton, B. J.; and Higginbotham, D. J. 1999. Effects of ngram order and training text size on word prediction. In *Proceedings of the RESNA '99 Annual Conference*, 52–54. RESNA Press.

Pauca, V. P., and Guy, R. T. 2012. Mobile apps for the greater good: A socially relevant approach to software engineering. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, 535–540. New York, NY, USA: ACM.

Tenenbaum, J. B.; Kemp, C.; Griffiths, T. L.; and Goodman, N. D. 2011. How to grow a mind: Statistics, structure, and abstraction. *Science* 331(6022):1279–1285.

Trnka, K., and McCoy, K. F. 2007. Corpus studies in word prediction. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '07, 195–202. New York, NY, USA: ACM.

Trnka, K., and McCoy, K. F. 2008. Adaptive word prediction for aac. In *2008 ISAAC Biennial Conference Proceedings*, ISAAC 2008.

Trnka, K.; Yarrington, D.; McCoy, K. F.; and Pennington, C. 2006. Topic modeling in fringe word prediction for acc. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, IUI '06, 276–278. New York, NY, USA: ACM.

Trnka, K.; McCaw, J.; Yarrington, D.; McCoy, K. F.; and Pennington, C. 2008. Word prediction and communication rate in aac. In *Proceedings of the IASTED International Conference on Telehealth/Assistive Technologies*, Telehealth/AT '08, 19–24. Anaheim, CA, USA: ACTA Press.

Trnka, K.; McCaw, J.; Yarrington, D.; McCoy, K. F.; and Pennington, C. 2009. User interaction with word prediction: The effects of prediction quality. *ACM Trans. Access. Comput.* 1(3):17:1–17:34.

Trnka, K.; Yarrington, D.; and McCoy, K. 2006. Topic modeling in fringe word prediction for aac. In *2006 ISAAC Biennial Conference Proceedings*, ISAAC 2006.

Trnka, K. 2008a. Adapting word prediction to subject matter without topic-labeled data. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '08, 315–316. New York, NY, USA: ACM.

Trnka, K. 2008b. Adaptive language modeling for word prediction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Student Research Workshop*, HLT-SRWS '08, 61–66. Stroudsburg, PA, USA: Association for Computational Linguistics.

Uchimoto, K.; Sekine, S.; and Isahara, H. 2002. Text generation from keywords. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, 1–7. Stroudsburg, PA, USA: Association for Computational Linguistics.

Vertanen, K., and Kristensson, P. O. 2011. The imagination of crowds: Conversational aac language modeling using crowdsourcing and large data sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, 700–711. Stroudsburg, PA, USA: Association for Computational Linguistics.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. Satzilla: Portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research* 32:565–606.