Topic Modeling in Fringe Word Prediction for AAC

Keith Trnka, Debra Yarrington, Kathleen McCoy
Computer Science Department
University of Delaware
Newark, DE 19716
{trnka,yarringt,mccoy}@cis.udel.edu

Christopher Pennington
AgoraNet, Inc.
314 E. Main St., Suite 1
Newark, DE 19711
penningt@agora-net.com

ABSTRACT

Word prediction can be used for enhancing the communication ability of persons with speech and language impairments. In this work, we explore two methods of adapting a language model to the topic of conversation, and apply these methods to the prediction of fringe words.

Keywords

Word prediction, keystroke savings, alternative and augmentative communication (AAC), topic modeling, language modeling

1. INTRODUCTION

Alternative and Augmentative Communication (AAC) is the field of research concerned with finding ways to help those with speech difficulties communicate more easily and completely. Today there are approximately 2 million people in the United States with some form of communication difficulty. One means to help ease communication is the use of an electronic communication device, which may have synthetic speech as output. However, one issue in using a communication device is that communication rate is generally slower than the common speaking rate. Whereas speaking rate is estimated at 180 words per minute (wpm) and experienced typists can manage 100 wpm, many AAC users' communication rates are lower than 15 wpm [3, 7, 21]. Thus one goal of developers is to find ways to increase the rate of communication, by making AAC devices easier to use and more intelligent.

This paper investigates the use of a word prediction system to speed communication rate by requiring fewer keystrokes to enter a word. In word prediction, we assume that the user selects characters one at a time. The system predicts full words that are likely to be desired, and provides them to the user for selection with one additional keystroke.

I want a	h	
	hundred	(F1)
	half	(F2)
	house	(F3)
	hard	(F4)
	home	(F5)

Figure 1: An example of what a word prediction interface might look like.

A word prediction system predicts the word currently being typed on the basis of what has already been typed. Suppose that the user wants to enter "I want a home in the country." After typing, "I want a h", they might see something like shown in Figure 1. The system has created a *prediction window* containing the five words that it thinks the user may be trying to type. In this example, the user can press F5 to complete the word "home" and the system will enter the word with a space afterwards. So in this example, the user needed 2 keystrokes to enter what would normally take 5 keystrokes, when the space is considered.[1]

The prediction list can vary in length, but most systems tend to use lists of length between five and seven. The prediction list can occur in-line or elsewhere on screen. For row-column scanning devices, the word list often appears in an extra row or column on the scanning grid.

It is difficult to judge how much word prediction can speed communication rate. Much of this determination is dependent on both the characteristics of the user, such as their physical and cognitive abilities, and characteristics of the user interface, such as where the prediction list is displayed and how a word in the list is selected. However, the prediction method can be evaluated separately from the rest of a word prediction system by simulating what a user would type in a conversation if he/she were taking full advantage of the prediction list. This theoretical evaluation measures the percentage of keystrokes that were saved by word prediction over typing out every character.

1.1 Core and Fringe Vocabulary

It is a well-known fact that a small portion of the English vocabulary comprises the majority of words used in a conversation. With this principle in mind, researchers in AAC have sought to improve communication rate by providing

[1]The list of predictions for this example was generated using our trigram baseline method, trained on the Switchboard corpus.

easy access to frequently-used words. This is the motivation behind what is called the *core vocabulary*. The core vocabulary is comprised of common closed-class words such as determiners and modals as well as common open-class words such as colors and days of the week. The portion of a person's vocabulary that isn't core vocabulary is deemed *fringe vocabulary*.

Many AAC devices provide separate interfaces for entry of core words as opposed to fringe words. Abbreviation expansion is a common method for entry of core words. Another common method is *semantic compaction* [1], which selects a word from the core vocabulary by entering a sequence of multi-meaning icons. One such device is the Pathfinder device of Prentke Romich Company [2], which allows entry of core words via icons and entry of fringe words via fringe word prediction. Because there are many established methods for speeding the user's access to core vocabulary, we seek to improve the state of the art of fringe word prediction.

Therefore, we look for language modeling methods which will more accurately predict fringe words. Our hypothesis is that taking the topic of a conversation into account will have a positive effect on fringe word prediction.

In this paper we first describe related work and give some background in statistical approaches to word prediction. We give an explanation and evaluation of our baseline word prediction system, then present the full details and evaluation of our approaches to integrating topic modeling into word prediction. For comparison purposes, we also discuss the impact of topic modeling to all-word prediction. Finally, we discuss future improvements to our language models and conclude.

2. RELATED WORK

2.1 Word Prediction

Clearly, we are not the first researchers to apply n-gram methods to word prediction in AAC [4, 5, 16, 24]. The PAL system was one of the earliest word prediction systems for AAC [24]. It used a simple unigram model to predict words. Boggess [4] experimented with a small collection of text from an AAC user. She tried correlating sentence position with words, basing the predictions off the location with in the sentence. She also evaluated a language model which can be thought of as a backoff trigram model with liberal pruning. Unlike later researchers, she employs a prediction window size of 20 and does not evaluate her model using keystroke savings. Rather, she records the percentage of times one keystroke is required to get the desired word in the prediction list, two keystrokes, etc. In contrast, Carlberger et. al. [5] describe the Profet system, a commercial word prediction system which uses n-gram methods. Their evaluations found that although keystrokes were saved, a speedup in communication rate was not always achieved. However, they found that word prediction greatly reduced physical fatigue and improved the grammaticality of users. Finally, Lesher et. al. [16] show the effect of increasing the size of the training material for an n-gram model as well as increasing from unigrams to bigrams to trigrams. They empirically show that a unigram model can be saturated with a reasonable amount of data, whereas a bigram and trigram model continue to improve with increasing training data, even at a training size of 3 million words. They also show that a bigram model yields a large improvement in keystroke savings over a unigram model (on the order of increasing from 47.2% to 54.7% keystroke savings). However, the trigram model only gave an additional 0.8% improvement. Their evaluations use a window size of 6.

Several researchers have integrated grammatical information into an n-gram word prediction system. Garay-Vitoria and González-Abascal [10] integrated a statistical chart parser into an n-gram based prediction method. They found that the addition of grammatical information improved keystroke savings by roughly 1-3% across window sizes, but presented most results using implicit rejection, a technique that assumes a user will select a word as soon as it appears in the list. If they do not choose a word in the list, but instead type another letter, all candidates in the list are omitted from further prediction. Fazly and Hirst [8] also investigated the use of syntax in word prediction, but used part-of-speech (POS) tagging rather than parsing. They combined a word bigram model with a trigram POS tag model and found that, at window size 5, keystroke savings was increased by approximately 1%. Copestake [7] also incorporated a POS tag model into an n-gram word prediction system, but found a 3-5% improvement in keystroke savings across window sizes 2-20.

Although there is a large existing research base of word prediction for AAC, the results of each researcher can't be adequately compared due to differences in evaluation methods. Firstly, there is little agreement between researchers on the size of the prediction list. Some researchers wait until one character has been entered to begin prediction and some begin predicting solely on the basis of context. However, the most difficult aspect of comparing results amongst different researchers is that there is no standard corpus to be used for evaluation.

Finally, word prediction isn't limited to the AAC community. Tegic's T9 Text Input [12] is an ambiguous keyboard for cell phones that uses word prediction. This greatly saves keystrokes over the traditional multi-tap method. Stocky et. al. [23] compared semantic prediction to various unigram models for use in word prediction for cell phones. They found that semantic prediction works very well on some content words, but poorly on others. Masui [20] also looked into word prediction to speed up text entry on small devices, but differs in that they sought to ease East Asian text entry on pen-based computers.

2.2 Topic Modeling

Like word prediction, topic modeling has been studied extensively. However, the vast majority of research in topic modeling can been to improve speech recognition. These researchers evaluate language models in terms of perplexity and word error rate (WER). Perplexity is a measure of how unpredictable a sequence of words are, and word error rate captures the frequency of speech recognition errors.

Bellegarda [2] sought to improve speech recognition by topic modeling using Latent Semantic Analysis (LSA). The model was trained on material from the Wall Street Journal. In testing, an exponential decay cache was maintained for each monologue. This cache was then applied to create a latent semantic representation of the current monologue's history. Probabilities from an n-gram model were then interpolated with unigram probabilities of the LSA represen-

[2]For more information, see http://www.prentrom.com/

tation of the cache and normalized. He achieved a 24.7% average reduction in perplexity and a 13.7% WER reduction using these techniques. He explored word-based clustering and reduced the word error rate further.

Mahajan et. al. [18] also used an implicit representation of topics to reduce test set perplexity on Wall Street Journal data, intended to improve speech recognition. They used the cosine similarity between a unigram vector of the current document's history and the unigram vector of each document in the training set to rank documents. The language models constructed from each of the top k documents were then interpolated to construct a topic model. With the addition of some optimizations, they reduce test set perplexity by 37.6%, but do not evaluate WER using a speech recognizer.

Seymore and Rosenfeld [22] and Chen et. al. [6] both apply topic modeling to speech recognition on the Broadcast News corpus. Their particular corpus was manually clustered into topics a priori. Seymore and Rosenfeld use a process much like Mahajan et. al., except that topics are defined explicitly. Chen et. al. instead identify the most likely topic of conversation and then use it to boost the probabilities of on-topic words and depress the probabilities of off-topic words, using an exponential model. Although their efforts reduce perplexity somewhat, word error rate is reduced very little, because perplexity does not very well with application performance.

Florian and Yarowsky [9] perform automatic hierarchical clustering on training data to create explicit topics. They use a process roughly similar to Mahajan et. al's to select the node in the hierarchy that is most appropriate, and then integrate the use of more general topic models into the backoff process. After some tuning of their model, they reduce perplexity on the Broadcast News Corpus by 10%.

Finally, Lesher and Rinkus [17] is the only work to our knowledge that applies topic modeling to word prediction. They use the Switchboard corpus and its topic labels. Their "Auto" model contains a separate trigram model for each of the 20 most frequent Switchboard topics. When a conversation is encountered in testing, the topic label of that conversation is checked and then the appropriate trigram model is used for that topic. Clearly, this in an unrealistic simulation, but it gives an idea of the upper boundary of this method of word prediction. That is, given that the topic of conversation is known, it shows what savings is possible. When compared to a language model of comparable size, the on-topic model wins by a margin of about 3%. However, a single trigram model constructed from all of the 20 topic models outperforms the Auto model by roughly 2%. However, they interpolate the larger trigram model with the Auto model to improve savings over the large trigram model by 1%.

3. METHODS

Like several of the aforementioned word prediction researchers, we use n-gram methods for language modeling. Our baseline word prediction methods use bigram and trigram-based n-gram models with backoff. These language models are used to rank all words that match what the user has typed. In our prior example (Figure 1), the language model ranks all words beginning with "h", and the user interface presents the most likely 5 words from the ranked words. However, if the user were trying to enter "hamburger" rather than "house", they would press "a" and the system would recreate the prediction window given the new prefix of the desired word.

To present the user with a list of possible words, the word prediction system needs to know all of the words in the language. The vocabulary is constructed by considering all words that occur in some training corpus. If a word being typed isn't in the vocabulary, it can't be predicted.

The second requirement is a statistical language model. The purpose of the language model is to compute the probability $P(word \mid history)$, where the history is an ordered list of all the words that have already been entered. [3] Given a vocabulary and a language model, the list of predictions is generated as follows: The vocabulary is first filtered to remove words that don't match the partially entered word. Then this list of candidates is sorted by $P(word \mid history)$. The top W words are presented to the user, where W is the prediction window size.

The remainder of this section is devoted to the construction of a statistical language model.

3.1 Corpus

Statistical approaches require a collection of text to construct a language model. Ideally, our corpus would be a large collection of conversations involving one or more people using an AAC system. Such a corpus is unavailable, so we follow [17] in using the Switchboard corpus, which is a collection of telephone conversations and their transcriptions.[4] Each conversation is assigned to one of 69 topics, and 8 conversations are unlabeled, which we assign to a separate topic. We have divided the corpus into two sections: one section to train the statistical language model and one section to evaluate the word prediction method. The training section contains a randomly pre-selected 2217 conversations and the testing section contains the remaining 221 conversations.

Because the corpus is a collection of transcribed conversations, it contains many speech repairs. Consider the example

> is there um an- is there a like a code of dress ...

However, a person in a textual conversation would write

> is there a code of dress ...

Hindle [11] categorized these sorts of self-corrections and integrated his solution into a parser. In short, his parser identified the words that are being corrected and the correction. The words being corrected are then removed for syntactic processing. Hindle's full set of editing rules was not practical for our purposes, so we implemented a subset of the rules: remove uh/um, exact repetitions, and repetitions in which the last word of the corrected part is abandoned, as in "an-". These editing rules bring the Switchboard conversations closer to what we envision an AAC user would type. This agrees with the preprocessing that [17] did according to personal communication with Dr. Lesher.

3.2 Baseline Language Model

[3] The history contains all words that have been entered, but the language model may choose to ignore the majority of them, which is the case for n-gram models.

[4] The Switchboard transcriptions were available from http://www.isip.msstate.edu/projects/switchboard/

N-grams have been shown to give better performance with larger amounts of training text and roughly better performance with larger n [16, 19]. For an introduction to n-grams, refer to [19] or [13]. The main weakness of n-grams is that they require substantial amounts of data. Using higher-order n-grams such as trigrams increases the problem of data sparseness. To combat this, we implement backoff with Good-Turing smoothing, the current best practice in statistical language modeling according to Manning and Schütze [19].[5]

3.2.1 Backoff

Backoff is applied here much like in [14]. The idea is that the probability will be determined by trigram probability if the trigram probability is defined, otherwise we'll try the bigram probability. If that's undefined, we'll try the unigram probability. The crux of backoff is adjusting these probabilities so that all probabilities sum to one.

In order to apply backoff, some probability must be removed from trigrams which were seen in the corpus. We use Good-Turing smoothing to do this so that each conditional trigram distribution will sum to less than one. This process is necessary to ensure that the probabilities obtained from the backoff model sum to one.

$$1 - \delta_{w_{-2},w_{-1}} = \sum_{w \in V} P(w \mid w_{-2}w_{-1})$$

Then the probability obtained by backoff is defined as

$$P'(w|w_{-2}w_{-1}) = \begin{cases} P(w|w_{-2}w_{-1}) & if P(w|w_{-2}w_{-1}) > 0 \\ \delta_{w_{-2},w_{-1}} \times & \\ \quad P'(w|w_{-1}) & otherwise \end{cases}$$

Similarly, we define the bigram backoff probability as:

$$P'(w|w_{-1}) = \begin{cases} P(w|w_{-1}) & if P(w|w_{-1}) > 0 \\ \delta_{w_{-1}} \times P'(w) & otherwise \end{cases}$$

3.2.2 Sentence-initial Model

The trigram backoff model just described is applied to all words in a sentence but the first word. The first word in the sentence is modeled using a special unigram model to capture the notion that sentences tend to begin with a small set of words such as determiners and discourse markers. If the probability of a given word is undefined, then we backoff to the normal unigram model, using the same type of backoff as for bigrams and trigrams. This differs slightly from the standard approach of creating a special start symbol which is included before the first word of every sentence. The difference reflects our intuition that only the first word of the sentence is affected by position within sentence, whereas the second word depends almost entirely upon the first word. Additionally, the traditional approach has the disadvantage of using more sparse data for prediction of the second word of a sentence.

3.3 Evaluation

In evaluating our models, we compare the number of keystrokes required for a user taking full advantage of our word prediction system to the number of keystrokes required to enter each character. We use *immediate prediction* for our evaluations, which allows use of the prediction list before the first character of a word has been entered. We assume that one keystroke is required to "speak" each turn of input and that a space is automatically inserted after a word is selected from the prediction list. We simultaneously consider prediction lists of size 1–10, but present a subset of our results due to space limitations.

[5]I'm not sure whether or not to include this, but our approach isn't the same as Katz' backoff, but similar.

Window size	Bigrams	Trigrams
3	54.7%	55.1%
5	58.6%	58.8%
6	59.8%	60.0%

Table 1: The keystroke savings achieved by a bigram backoff model and trigram backoff model are shown for window size 3, 5, and 6.

$$KS = \frac{keys_{normal} - keys_{withprediction}}{keys_{normal}} * 100\%$$

Because we are interested in the prediction of fringe words, the majority of our evaluations are measured on fringe words only. Core words are excluded from the list of predictions.[6] The particular core vocabulary we chose is available from the AAC Centers at the University of Nebraska at Lincoln, available from http://aac.unl.edu/. We used the "Young Adult Conversation Regular" core vocabulary list, as it is the most similar to the type of conversations in the Switchboard corpus.

The results of word prediction using a bigram backoff model and a trigram backoff model are shown in Table 1. Just like Lesher et. al. [16] found, trigrams offer only a small improvement in keystroke savings over bigrams. This is most likely due to the relatively small size of our training set (2.6 million words). For reference, the maximum possible keystroke savings on this test set is 82.5%. Clearly, there is room for improvement.

4. TOPIC MODELING

The goal of topic modeling is to identify the current topic of conversation, then increase the probability of related words and decrease the probability of unrelated words. Some words will be unaffected by topic modeling, such as function words, which are used similarly in all topics. It is for this reason that we chose to improve fringe word prediction with topic modeling: we feel that topic modeling specifically improves fringe word prediction.

4.1 Topic Representation

Researchers are consistent in representing a topic by creating a collection of representative text of the topic. However, researchers differ on the best way to organize a collection of topics. Some researchers have created a hierarchical collection of topics [9], while others have created a disjoint set of topics [18, 2, 22]. The primary advantage of a hierarchical approach is that a more general topic can be selected when the topic of conversation is difficult to identify, and can be very specific when the topic of conversation is clear. However, as Mahajan et. al. point out [18], a conversation may simultaneously discuss multiple topics. The set approach is more amenable to allowing the selection of multiple topics. We feel that the primary lure of a hierarchical approach, the

[6]But see Section 5 for an evaluation on the full vocabulary.

ability to generalize, can be captured in the set approach as well, by giving varying weight to all topics and not just the most likely topic.

4.2 Topic Identification

The current topic of conversation must be identified from the part of the conversation that has taken place so far, and updated periodically in the conversation. Thus, we must devise a representation for a partial conversation for assessing the similarity of the conversation to each topic. We follow other researchers in maintaining something like a unigram model of the current conversation and applying document similarity measures to identify appropriate topics.

In representing the conversation so far, we choose to implement an exponentially decayed cache, like [2], using TF-IDF values rather than raw frequencies. This follows the work of Mahajan et. al. [18] in considering the inverse document frequency of a word to proportional to its utility in identifying the current topic. The algorithm proceeds as follows: When a word is added to the cache, first all words that have occurred previously are decayed. This is done by multiplying their weights by λ, a constant between 0 and 1. For our experiment, we followed Bellegarda in using $\lambda = .975$. A high value of λ favors conversations that change topic slowly, and a low value favors conversations that change topics often. Secondly, the new word is added with it's IDF as the weight instead of 1. Because our approach is for topic identification, we ignore words that occur in 85% or more of the topics, with the intuition that such words are irrelevant to selection of topic. If such words were included, they might lead to artifacts in identifying the current topic due to the limited size of data for each topic.

As a step to convert our model of the current conversation to a model of the current topic, we compute the document similarity between the cache and the unigram model for each topic. Many different measures of similarity have been used previously in literature. See [15] for a comparison in the context of smoothing, using terms that appear in similar contexts. However, we chose to use the cosine metric, following [9].

4.3 Topic Application

Given that we have computed a similarity score between each topic and the current conversation, there are two main variations on how to construct a new language model. Mahajan et. al. [18] implemented a k-nearest solution, constructing the topic model from the most similar k topics. Each topic's language model was weighted equally for their experiments. Instead, we chose to follow Florian and Yarowsky's approach [9]. They expand the probability for a word (w) given a history (h) as follows:

$$P(w \mid h) = \sum_{t \in topics} P(t \mid h) * P(w \mid t, h)$$

$P(w \mid t, h)$ is simply the probability of w taken from the language model constructed for topic t. The probability of the topic is estimated as follows:

$$P(t \mid h) \approx \frac{S(t, h)}{\sum_{t' \in topics} S(t', h)}$$

where $S(t, h)$ is the cosine similarity of the topic to the current part of the conversation.

4.4 Practical Considerations

Although we are primarily interested in investigating whether or not topic modeling improves keystroke savings in fringe word prediction, a number of practical considerations have influenced our approach.

4.4.1 Smoothing and Topic Modeling

The backoff model requires that smoothing be performed on each conditional distribution. Smoothing can be applied at two different points in processing with topic modeling. Firstly, each individual topic language model can be smoothed. The alternative is to smooth the interpolated topic model. As there may be many topics, it's likely that each topic model will be rather sparse, whereas the interpolated model will be less sparse. If we apply smoothing before interpolation, bigrams and trigrams may be given less weight than is appropriate for the interpolated model. It is for this reason that we choose to apply smoothing after interpolation. Normally, this would lead to increased time of execution, but our implementation only smooths distributions that are reached in testing as they are reached. However, smoothing requires frequencies rather than probabilities. To solve this, we approximate the interpolated topic model by using frequencies in the interpolation equation instead of probabilities.[7] However, the interpolated frequencies are not whole numbers do to the interpolation process. We address this problem by taking the ceiling of each frequency before smoothing.

4.4.2 Number of Topics and k-nearest Topics

We had originally intended to follow [18] in abandoning the notion of an explicit topic. Their approach treats each document in the training set as it's own topic, and relies upon the interpolation model to select the relevant topics and make appropriate generalizations. This avoids the problem of either finding a corpus annotated for topic or the problem of performing appropriate automatic topic clustering. Unfortunately, our preliminary tests revealed that interpolation of a topic model from about 2,200 smaller topic language models was too computationally demanding. Instead of treating each conversation as a topic, we used the topic assignments given in the Switchboard corpus. This reduces the number of topics to 70.

4.4.3 Dynamic Topic Adaptation

As we intend our research to be used for practical word prediction systems, we designed our topic model as practically as possible. That means, unlike Mahajan et. al. [18], we only have access to the parts of the conversation that have been already said. Their approach assumes that the first 100 words of a conversation are available a priori. Also, unlike [17] , we do not assume to know the topic of a conversation ahead of time in testing. Their investigation used Switchboard, like ours, but read the topic number of each conversation to select the appropriate topic in testing.

However, dynamically re-interpolating the topic model is computationally expensive. Although we re-interpolate the topic modeling as the conversation proceeds, we only re-

[7]The problem which this causes is that topics are now not only weighted by their topical similarity, but by their size, as topics with higher frequencies will tend to dominate the model. However, models with higher frequencies are also more reliable.

Window size	Bigrams	Topic (A)	Improvement
3	54.7%	56.4%	1.7%
5	58.6%	60.2%	1.6%
6	59.8%	61.4%	1.6%

Table 2: The keystroke savings of the bigram baseline is shown compared to the topic model (Method A).

Window size	Trigrams	Topic (A)	Improvement
3	55.1%	56.4%	1.3%
5	58.8%	60.2%	1.4%
6	60	61.4%	1.4%

Table 3: The keystroke savings of the trigram baseline is shown compared to the topic model (Method A).

compute the model after every second turn to mitigate the computational cost of interpolation.

4.5 Method A

Our first method of topic modeling is most similar in spirit to the work of Mahajan et. al. [18] and Florian and Yarowsky [9]. In training, a bigram model is computed for each topic in Switchboard [8]. In testing, the cache representation of the current conversation is compared against the unigram representation of each topic and similarity scores are computed. The similarity scores are then used to weight the frequencies obtained from each topic in a linear interpolation. Then this interpolated bigram model is used to compute the probabilities used for word prediction. The evaluation of this method is shown in Tables 2 and 3.

Topic modeling shows a sizable improvement over the the bigram baseline: 1.6% – 1.7%. We've included the comparison to a bigram baseline because it is the most natural baseline in terms of language understanding. However, a trigram baseline is also a natural comparison when considering that it can run with the same or less computational resources than topic modeling. When compared against the trigram baseline, the topic model gives 1.3% – 1.4% improvement.

4.6 Method B

Our second method of topic modeling is more similar to the work of Bellegarda [2]. Like Bellegarda, we compute topic-dependent unigram probabilities. These topic-dependent probabilities are combined geometrically with a trigram backoff model, as shown below. *norm* is a normalization factor.

$$P(w \mid h) = \frac{P_{trigram}(w \mid h) * P_{topic}(w)^{\alpha}}{norm}$$

As an optimization, we omit the normalization. This is possible because the probabilities are only used to rank candidate words.

The main deviation from Bellegarda's model is the inclusion of a tuning parameter, α. For his work, an un-tuned model was sufficient. However, preliminary simulations on a 2-conversation test set showed that tuning was necessary for this method to be an improvement over a trigram baseline. After tuning manually on this set, we found that $\alpha = .15$ was a good weight. This weight allows the topic's vocabulary to have an effect, but the vocabulary preference won't override the context of the current prediction. The results of this method are shown in Table 4.

[8]We used bigram models instead of trigram models due to computational limitations.

Window size	Trigrams	Topic (B)	Improvement
3	55.1%	55.4%	0.3%
5	58.8%	59.1%	0.3%
6	60.0%	60.3%	0.3%

Table 4: The keystroke savings of the trigram baseline is compared to the topic model (Method B).

Window size	Method A	Method B	Difference
1	43.1%	42.5%	0.6%
2	52.3%	51.4%	0.9%
3	56.4%	55.4%	1.0%
4	58.7%	57.7%	1.0%
5	60.2%	59.1%	1.1%
6	61.4%	60.3%	1.1%
7	62.2%	61.1%	1.1%
8	62.9%	61.8%	1.1%
9	63.5%	62.3%	1.2%
10	64.0%	62.8%	1.2%

Table 5: The keystroke savings of Method A and B are compared. The difference column shows the improvement of Method A over Method B.

Method B is an improvement over a trigram baseline, but only a minor improvement. We feel that the problem is that a low α value was necessary to avoid overriding the preference due to context, but that it also reduced the ability of the overall model to adapt to a particular topic. This might be improved by a model designed specifically to address this issue.

4.7 Comparison

For a side-by-side comparison of the two methods of topic modeling, see Table 5. Method A offers an additional 1% or more keystroke savings over Method B for most window sizes. This is due to the low weight of the tuning parameter for Method B. However, as previously mentioned, the low weight was necessary. Additionally, notice that Method A becomes comparatively better as the window size is increased. The explanation isn't that Method A becomes better at higher window sizes, rather, Method B is more suited for small window sizes. Recall that Method B includes a trigram backoff model. This trigram model can be thought of as a stronger source of knowledge than the interpolated bigram model. Because of this, when the trigram history exists in the language model, Method B's predictions are more accurate. However, because the trigram model is sparse, it can only contribute to the top few predictions. Thus, it has a much greater effect on the top few window sizes. The same trend can be seen without topic modeling in Table 1.

For real world systems, however, absolute performance is not the only factor. The computational demands of each approach are often considered when selecting a practical solution. The trigram baseline processed at 1,325 words per minute (wpm). Method A processed conversations in test-

ing at 32 wpm and Method B processed 1,267 words per minute. Method B uses barely more processing time than the trigram baseline model. However, Method A runs a great deal slower. Although Method A still predicts fast enough on average to not limit the communication rate of many AAC users, this speed is only an average. The processing time is not uniform - more processing time is used to generate the list when zero or one characters have been entered than when two characters have been entered. On the other hand, the average speed of Method B and the trigram baseline is fast enough to keep up with even the most experienced typist. In the end, choice of which method to use depends on the situation. If computational resources are limited, Method B is appropriate as an improvement over a trigram baseline. However, if computational resources are substantial or if the user is very limited in their communication rate, Method A is more appropriate.

5. ALL-WORD PREDICTION

Although we have sought to improve fringe word prediction by dynamic topic adaptation, some AAC users use word prediction exclusively to communicate. For these users and the corresponding word prediction systems, it is useful to evaluate the impact of topic modeling on all words rather than fringe words only. The results of this evaluation are shown in Table 6.

As seen before, the trigram-based model compares better against Method A at smaller window sizes. However, unlike with fringe word prediction, at window sizes 1-3 the trigram model is better than Method A. Method B shows a relatively steady improvement in keystroke savings over the trigram baseline across all window sizes except 1. At window size 1, the prediction is most likely determined by very strong evidence from the trigram model. That strong evidence is unlikely to be out-voted by the topic model with $\alpha = .15$. Unlike fringe word prediction, Method B offers equal or better savings than Method A for window sizes 1-6, and worse for window sizes 7-10. Again, this is due to the strong evidence of trigrams. However, because most common word prediction systems use window size 7 or smaller, Method B is more suitable for all-word prediction.

It is also instructive to compare the keystroke savings of all-word prediction to fringe word prediction. The keystroke savings of word prediction on fringe words only (Tables 5 and 1) is better than the savings on all words (Table 6) across all window sizes. Core words are excluded from predictions for fringe word prediction, thus, fringe word prediction is a specialized prediction method. This follows the general trend in NLP that a specialized method on a restricted problem often outperforms a general method on a more general problem.

The only other work to use word prediction in conjunction with the Switchboard corpus is [17]. They used a trigram model with a window size of 6 to evaluate topic modeling. They did not include a "speak key" in their simulations, which we have found inflates results by 1% on our testing section of the Switchboard corpus at a window size of 6. In addition, they did not implement dynamic topic modeling, but instead assumed to know the topic of each conversation in testing a priori. Their trigram baseline model achieves 57.81% keystroke savings and their topic model, when interpolated with the trigram model, achieves 58.74% keystroke savings. By comparison, our trigram baseline saves 59.1% of keystrokes, and our dynamic topic models save 59.3% of keystrokes. Although the conditions under which they have evaluated their methods differ from ours, this is the best comparison to existing research in word prediction that we can make.

6. FUTURE WORK

Although our long term plan includes the development of language models that don't consider the topic of conversation, we feel that there is still some room left for improvement in topic modeling. Firstly, Method B might be improved by incorporating it more closely with the trigram model. Secondly, we have shown that a specialized fringe word prediction model performs better on it's task than a general-purpose model on a general-purpose task. Therefore, we would like to investigate the use of a fringe-specialized model and a core-specialized model using class-based smoothing.

The idea of language modeling is to understand which word or words to use in a particular situation. Clearly, n-gram methods do not capture the full extent of why a person chooses to say one word as opposed to another. Topic modeling helps to model more of what isn't captured by n-grams, namely, long-distance and variable-distance dependencies between words. However, much work remains in adequately accounting for the particular use of words in conversation. One of the several alternative approaches we intend to investigate is style modeling. While topic modeling will primarily influence the vocabulary usage, we envision that a style model would primarily influence higher-order n-grams in a similar fashion to topic modeling. Other similar language modeling improvements include filtering the list of predictions for syntactic correctness and the integration of compound words as a single unit into both the predictions as well as the context for prediction.

Finally, our long term goal is to incorporate several different and complimentary language models into a single word prediction engine. It is our intuition that word choice in natural language is determined by several different factors. For example, terminology is clearly affected by topic. However, choice amongst synonyms is largely determined by user preference and style. Thus, our long term plan is to create several independent word prediction methods that capture complimentary aspects of language and then develop a model to combine the separate, specialized language models.

7. CONCLUSIONS

Topic modeling can be implemented in many different ways. We've demonstrated two such methods for topic modeling: one for computationally limited devices and another for computationally rich devices. Both methods show a clear improvement over a trigram model with backoff. Before the advent of word prediction, a user would've pressed 6.4 keys per fringe word on average. Now, with topic modeling for word prediction, only 2.5 keys per word are required.

8. ACKNOWLEDGMENTS

We would like to thank the US Department of Education for funding this research under grant H113G040051, under the National Institute on Disability and Rehabilitation Research program. We would also like to thank Dr. Gregory Lesher for correspondence regarding his work and Dr. David Saunders for lending us a compute server.

Window size	Bigrams	Trigrams	Method A	Method B	MA over Trigrams	MB over Trigrams
1	39.5%	40.7%	40.4%	40.7%	-0.3%	0
3	52.7%	53.5%	53.5%	53.7%	0	0.2%
5	57.1%	57.7%	57.9%	57.9%	0.2%	0.2%
6	58.6%	59.1%	59.3%	59.3%	0.2%	0.2%
7	59.7%	60.1%	60.5%	60.4%	0.4%	0.2%

Table 6: The keystroke savings of word prediction on all words is shown for all prediction methods. The improvement of Method A (MA) over the trigram baseline is shown, as well as the improvement of Method B (MB).

9. REFERENCES

[1] B. Baker. Minspeak. *Byte*, pages 186–202, 1982.

[2] J. Bellegarda. Large vocabulary speech recognition with multispan language models. *IEEE Trans. On Speech and Audio Processing*, 8(1), 2000.

[3] D. R. Beukelman and P. Mirenda. *Augmentative and alternative communication: Management of severe communication disorders in children and adults*. P.H. Brookes Pub. Co., 1998.

[4] L. Boggess. Two simple prediction algorithms to facilitate text production. In *Proceedings of the second conference on Applied natural language processing*, pages 33–40, Morristown, NJ, USA, 1988. Association for Computational Linguistics.

[5] A. Carlberger, J. Carlberger, T. Magnuson, M. S. Hunnicutt, S. Palazuelos-Cagigas, and S. A. Navarro. Profet, a new generation of word prediction: An evaluation study. In *Proceedings of Natural Language Processing for Communication Aids*, 1997.

[6] S. Chen, K. Seymore, and R. Rosenfeld. Topic adaptation for language modeling using unnormalized exponential models. In *Proc. Int'l Conf. on Acoustics, Speech and Signal Processing*, 1998.

[7] A. Copestake. Augmented and alternative nlp techniques for augmentative and alternative communication. In *Proceedings of the ACL workshop on Natural Language Processing for Communication Aids*, 1997.

[8] A. Fazly and G. Hirst. Testing the efficacy of part-of-speech information in word completion. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.

[9] R. Florian and D. Yarowsky. Dynamic nonlocal language modeling via hierarchical topic-based adaptation. In *Proceedings of ACL'99*, pages 167–174, 1999.

[10] N. Garay-Vitoria and J. González-Abascal. Intelligent word-prediction to enhance text input rate. In *Proceedings of the second international conference on Intelligent User Interfaces*, 1997.

[11] D. Hindle. Deterministic parsing of syntactic non-fluencies. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, 1983.

[12] C. L. James and K. M. Reischel. Text input for mobile devices: comparing model prediction to actual performance. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 365–371, New York, NY, USA, 2001. ACM Press.

[13] D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, 2000.

[14] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. On Acoustics, Speech, and Signal Processing*, 35(1), 1981.

[15] L. Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[16] G. Lesher, B. Moulton, and J. Higgonbotham. Effects of ngram order and training text size on word prediction. In *Proceedings of the RESNA '99 Annual Conference*, 1999.

[17] G. Lesher and G. Rinkus. Domain-specific word prediction for augmentative communication. In *Proceedings of the RESNA '02 Annual Conference*, 2002.

[18] M. Mahajan, D. Beeferman, and X. D. Huang. Improved topic-dependent language modeling using information retrieval techniques. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1999.

[19] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2000.

[20] T. Masui. An efficient text input method for pen-based computers. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 328–335, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[21] A. Newell, S. Langer, and M. Hickey. The rôle of natural language processing in alternative and augmentative communication. *Natural Language Engineering*, 4(1):1–16, 1996.

[22] K. Seymore and R. Rosenfeld. Using story topics for language model adaptation. In *Proceedings of Eurospeech '97*, pages 1987–1990, Rhodes, Greece, 1997.

[23] T. Stocky, A. Faaborg, and H. Lieberman. A commonsense approach to predictive text entry. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1163–1166, New York, NY, USA, 2004. ACM Press.

[24] A. L. Swiffin, J. A. Pickering, J. L. Arnott, and A. F. Newell. Pal: An effort efficient portable communication aid and keyboard emulator. In *Proceedings of the 8th Annual Conference on Rehabilitation Techonology*, pages 197–199, 1985.