Word Prediction Techniques for User Adaptation and Sparse Data Mitigation

Ph.D. Thesis Proposal

Keith Trnka
trnka@cis.udel.edu

Department of Computer and Information Sciences
University of Delaware
Newark, DE

March 19, 2008

Contents

Abstract

In the United States alone, it is estimated that approximately two million people suffer from a speech disability severe enough to create a difficulty in being understood [9]. This affects individuals in all aspects of their lives, from education to the workplace and in their personal lives. High-tech AAC devices are electronic devices that take letter and word input and produce speech output. However, communication rate with AAC devices is often below 10 words per minute [63], compared to the common 130-200 words per minute speech rate of speaking people. Word prediction is a technique to speed up communication rate that relies on the letters and words that the user has entered so far to suggest words that the user is in the process of typing, reducing the amount of input to produce utterances. Word prediction in modern devices utilize a trigram model and some form of recency promotion at best. However, one of the lamented weaknesses of ngram models is their sensitivity to the training data. The problem of utilizing ngram models for conversational AAC usage is that no substantial corpora of AAC text exist (much less conversational AAC text).

In this work, we address the corpus problem in two ways. Firstly, we will design language models that more fully utilize the available information in text to predict words, such as topic, style, and the words used so far in the text. Secondly, we will design language models in the mindset of sparse data. We must be able to take full advantage of small amounts of data, such as the text available in the current conversation. We plan to simultaneously adapt to topic and style by integrating the adaptations into a single part-of-speech ngram model. We will also apply cache-based adaptations in the part-of-speech model, which we expect to both model local fluctuations in word usage and account for new vocabulary, and also help compensate for the limited amount of relevant training data.

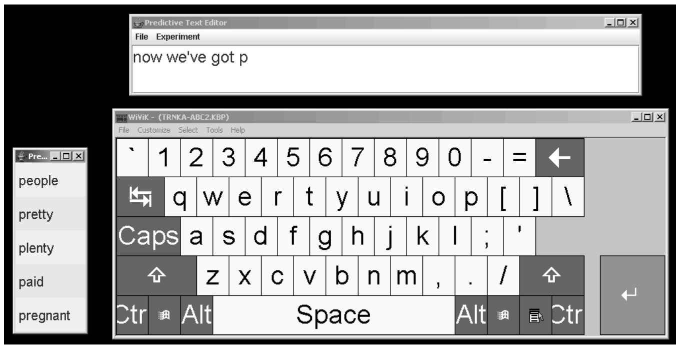
We have created a topic adapted language model that utilizes the full training data, boosting relevant portions and depressing irrelevant portions. We have demonstrated that topic modeling can significantly increase keystroke savings for both traditional testing [77], but also when tested on text from other domains [75]. We have also sought to address the problem of annotated topics through fine-grained modeling and found that it is also a significant improvement over a baseline ngram model.

We expect that this completed work will contribute a language model that is explicitly adapted to the style of discourse (both in general and for word prediction). Our application of Kuhn and de Mori's cache model [47] to word prediction will also be a new contribution. Our approach to integrating the three adaptive components will also further the field, as well as the evaluation methods for both the individual and combined models.

1 Introduction

In the United States alone, it is estimated that approximately two million people suffer from a speech disability severe enough to create a difficulty in being understood [9]. This affects individuals in all aspects of their lives, from education to the workplace and in their personal lives. An assortment of disorders are responsible for limiting speech, including Cerebral Palsy (CP), Amyotrophic Lateral Sclerosis (ALS), traumatic brain injury (TBI), muscular dystrophies, and congenital deafness. The field of Augmentative and Alternative Communication (AAC) seeks to help minimize the effects of such disorders on communication to allow all individuals to communicate with equal success. Unfortunately, disorders that limit or inhibit a person's use of speech often affect motor control as well, making the development of AAC solutions challenging. High-tech AAC devices are electronic devices that often address the dual issue of speech impairments and reduced motor control. They allow an individual to enter words and typically vocalize the text on the user's behalf. Speech Language Pathologists and AAC device vendors work with users to set up a usable physical interface for each device based on the motor control of each AAC user. PRC's Pathfinder, Dynavox's DynaWrite, and Saltillo's ChatPC are examples of available electronic AAC devices.

However, communication rate with AAC devices is often below 10 words per minute [63], compared to the common 130-200 words per minute speech rate of speaking people. This causes a communication rate gap between AAC communicators and traditional communicators, which can lead to social problems. For example, when an AAC user is conversing with a speaking person, sometimes the speaking person will naïvely attempt to assist the AAC user by completing every sentence for them and phrasing utterances to minimize the amount of typing required by the AAC user. However, such efforts cause the speaking person to dominate the conversation, and make it difficult for the AAC user to communicate what they originally wanted to. In addition to the effects of different communication rates, constant usage of an AAC device throughout the day can contribute to physical fatigue (which in turn can affect communication rate). AAC devices can address these two problems by reducing the amount of input required to produce utterances. Word prediction is a technique that relies on the letters and words that the user has entered so far to suggest words that the user is in the process of typing. While the user is typing an utterance letter-by-letter, the system continuously provides potential completions of the current word. The user may select a word from the list for one keystroke and the system will complete the word for them and enter a space afterwards. An example screenshot of word prediction from our user studies is shown in Figure 1. Although our example shows the completion of a word, a word prediction system may provide candidate words before any letters have been typed. Word prediction systems have the potential to save significant time and effort in entering words and many commercial AAC systems include word prediction for this reason, such as PRC's Pathfinder, Dynavox' DynaWrite, and Saltillo's ChatPC. The list of predicted words is generated using statistical


now we've got p

people
pretty
plenty
paid
pregnant

Figure 1: Screenshot of our touch screen-based user study word prediction system using the WordQ onscreen keyboard.

language models. However, many devices such as the Pathfinder primarily use unigrams only, though advanced word prediction software such as WordQ seems to use bigrams and trigrams.

At best, modern devices utilize a trigram model and some form of recency promotion. However, these models only partially account for the constraints a human would use (e.g., Lesher et al. [54] found that humans were better at word prediction than ngram models). The use of a basic language model results in many predictions that are not appropriate for the current conversation. For example, suppose a user is typing "vowel" in "I would like to buy a vowel" in Figure 2, using a trigram model with backoff trained on Switchboard data. One can imagine that a user seeking to enter the word "vowel" might become very distracted by looking at the predictions. Predictions such as "vacation", "voice", and "volunteer" are likely to be somewhat jarring, especially if "volunteer" is interpreted as a head noun. Inappropriate predictions lead to two major problems for users. While typing out the desired words, if a user is spending their time scanning the prediction list after every character and the desired word rarely appears, they could communicate much more quickly if they simply ignored the predictions and typed every word out. Trnka et al. [76] shows that when users are presented with a poor word prediction system, they often adopt the strategy of scanning the list less frequently, causing them to sometimes miss the desired word when it does appear, and therefore lose some of the potential speedup of the system. Conversely, when the predictions offered by the system are better, users utilize the predictions more often and show a communication rate increase due to both the better system but also due to increased user trust in the suggested words. In addition to the complication of user trust and how often they scan the predictions, researchers have suggested that the "jarring" aspect of inappropriate predictions may further slow down users' communication rates. Therefore, a system with more syntactically and semantically appropriate suggestions would likely avoid this additional cognitive dissonance. Our solution to these problems is to enhance statistical language models in word prediction to account for linguistic knowledge that traditional ngram models lack, including the topic and style of discourse.

I would like to buy a	
lot	(F1)
little	(F2)
good	(F3)
couple	(F4)
big	(F5)

(a) No letters

buy a v	
very	(F1)
variety	(F2)
vacation	(F3)
van	(F4)
video	(F5)

(b) 'v' entered

buy a vo	
voice	(F1)
voluntary	(F2)
volunteer	(F3)
vote	(F4)
volume	(F5)

(c) 'vo' entered

Figure 2: Example of trying to type "vowel" when predictions are inappropriate.

Bigram and trigram models are typical for advanced word prediction systems, due to their surprising robustness. However, one of the lamented weaknesses of ngram models is their sensitivity to the training data. Ngram models require substantial training data to be accurate, and as the order of the model is increased, a much larger amount of data is necessary for it to be useful. For example, Lesher et al. [55] demonstrate that bigram and trigram models are not saturated[1] even when trained on 3 million words, while a unigram model is nearly saturated. In addition to the problem of needing substantial amounts of training text to build a reasonable model, ngram models are typically very sensitive to the difference between training and testing/user data. An ngram model which is trained on newspaper text and evaluated on conversational language will likely perform very poorly compared to a model trained and tested on text of the same topic and style. We have demonstrated the domain sensitivity of trigram models for word prediction in [75]. Wandmacher and Antoine [83] have demonstrated the domain sensitivity of 4-gram models for word prediction in French.

The problem of utilizing ngram models for conversational AAC usage is that no substantial corpora of AAC text exist (much less *conversational* AAC text). The little available text from AAC users is largely written text from academic writing, magazine articles, book chapters, and public archives of electronic mailing lists. However, in addition to the problem of non-conversational text, some of these sources have been modified by editors. Even beyond the problem of AAC text, the majority of non-AAC corpora are written text. Corpora of spoken English are typically much smaller in size compared to written corpora. For example, the British National Corpus (BNC) is intended to be a wide cross-section of British language, and contains 89.39 million words of written text, but

[1]A language model that is saturated does not show significant improvements with more training data.

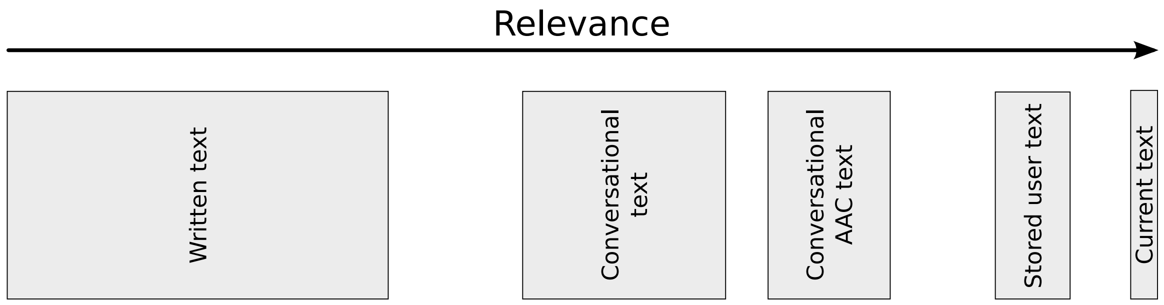


Figure 3: The most relevant text available is often the smallest, while the largest corpora are often the least relevant for AAC word prediction.

only 10.58 million words of spoken language. Similarly, the American National Corpus contains 22.39 million words of written text, but only 3.86 million words of spoken language. Even ignoring corpora that seek to sample a language, Switchboard contains about 2.88 million words of spoken language, whereas TIPSTER, released at about the same time, contains about 448 million words of written text. The issue of appropriate corpora affects both training and testing language models, but in different ways. If we were even able to have just appropriate testing data and not training data, we would be able to obtain a reasonable expectation of how our research affects actual AAC users. The problem of corpora is that similarity and availability are inversely related, illustrated in Figure 3. At one extreme, a very large amount of formal written English is available to use, however, it is very dissimilar from conversational AAC text. At the other extreme, logged text from the current conversation of the AAC user is the most highly related text, but it is extremely sparse. In this work, we will work around the corpus problem in two ways. Firstly, we will design language models that more fully utilize the linguistic information in text to predict words, such as topic and style. Secondly, we will design language models in the mindset of sparse data. We must be able to take full advantage of small amounts of data, such as the text available in the current conversation.

We propose adaptive language modeling techniques designed for sparse data as the solution to the lack of AAC user data and also as a means of more fully utilizing the available linguistic information. In order to make predictions more reasonable, we propose to adapt to the topic and style of conversation. Not only should this benefit the user more than traditional ngram models, but also this adaptation should help compensate for differences between training and testing data by focusing on the most relevant portions of the training data first. We plan to simultaneously adapt to topic and style by integrating the adaptations into a single part-of-speech ngram model. The adaptations will operate by first judging the topical and stylistic relevance of each portion of training data, and then weighting those portions higher in a re-training approach. This model will not only allow for topic and style adaptation, but also is better suited for limited training data size. We will also apply cache-based adaptations in the part-of-speech model, which we expect to both model local fluctuations in word usage and account for new vocabulary, and also help compensate for the limited amount of relevant training data.

1.1 Background

There are several related issues in devices for Augmentative and Alternative Communication (AAC). Several researchers have questioned the benefit of word prediction for increasing communication rate, due to the additional cognitive load of using word prediction. In their studies, they found that word prediction gave marginal improvements to communication rate at best, and lowered communication rate at worst [81, 46, 3]. Suggested components of the additional cognitive load include scanning the list to check if the desired word appears, recognizing the word in the list if it appears, and any additional physical movement needed to select words from the list of predictions. Previous researchers forced users to scan the predictions in a very specific strategy, commonly scanning after every letter entered. In contrast, users of text entry systems in general naturally develop their own strategies for optimizing communication rate. In our study [76], we allow users to develop their own strategy of usage to optimize communication and allow them to either use or ignore word prediction information as they wish. We not only found that we could measure the increased cognitive load due to word prediction (lower input rate), but that communication rate increased despite the cognitive load. We also found that most users did not fully utilize the predictions, but had much higher prediction utilization for better language models.

Several interface factors will affect the utility of a word prediction system. For example, one of the often cited factors in predictive interfaces is the number of elements in the prediction window [30]. A larger number of predictions will give higher theoretical keystroke savings. However, each additional prediction gives less and less benefit to the user but adds at least a linear time increase in scanning the list. Therefore, the number of predictions should be limited to a practical number. A list of 5 words seems to be the most common evaluation by researchers that is also manageable to scan [26, 59]. The orientation of the prediction window is also acknowledged to affect the usability of word prediction. A vertical list is agreed to be easiest for users to scan [30].

Development of word prediction for AAC devices is also faced with difficulties due to the large variety in device usage. People use their devices not just for face-to-face communication, but also communicating over the phone, for pre-planned speeches, writing emails, and writing formal written text. Our primary focus is conversational speech, as the real-time nature of the medium pressures users to communicate quickly (which can cause more fatigue than casually writing an email). However, we will include written text in our evaluations for devices that may potentially apply our research to written forms of communication.

One of the traditional features of commercial AAC devices is to allow a user to access some static vocabulary via single buttons or a sequence of 2–3 buttons. Many common devices such as PRC's Pathfinder use a form of encoding words with icon sequences called Minspeak. The Pathfinder PC emulator is shown in Figure 4 in spelling mode, but in Minspeak mode, the buttons' meaning is tied to the icon rather than the letter. For example, a user typing "fettuccine" with Minspeak might press the apple icon for food, then the medical icon (the snakes supposedly look like pasta), then the dog icon because "fettuccine" sounds like "fetch". In this way, one can view Minspeak as a three-layered hierarchical access to the vocabulary, but where icons or pairs of icons represent a category, and may be used in multiple different ways (e.g., the category of the icon's word, the color/shape of the picture, the sound of the word). Other devices often provide quick access to common words using either direct buttons or a hierarchical display. The subset of a user's vocabulary encoded in Minspeak or a hierarchical display is called the *core vocabulary*, and is intended to represent a subset of English that is common for a user's daily life with a focus on providing the user with basic ways to communicate many things. For example, domain-specific words such as "linguistics" or connotation-laden words such as "junker" are unlikely to be included in the core vocabulary of AAC devices. These kinds of words (those that are not core) are called *fringe vocabulary*. Instead, the AAC user might choose to say "word researcher" or "bad car" to take advantage of encoding system for core words. If the user wants to use a fringe word, they must enter spelling mode and type the word letter-by-letter. Even devices that rely on a heavily optimized core vocabulary interface such as PRC's Pathfinder provide the user with word prediction for spelling mode. Because access to core vocabulary is heavily optimized but access to fringe vocabulary often uses relatively poor word prediction, we have chosen to focus our research and evaluation on fringe word prediction.

Language modeling in word prediction is similar to language modeling in other domains, where we abstractly denote a language model as a conditional probability distribution over words: $P(w \mid h)$. The abstract part of this notation is the history h. An ngram model interprets h as the most recent few words (the number of words is the order of the model) and looks up the conditional probability table for the next word. In contrast, a cache model may interpret h as the training text for the ngram model. Given some language model $P(w \mid h)$, word prediction computes the probability of all words in the vocabulary and then sorts them in descending order, taking the top W and preserving their order to create the prediction window.

Word prediction is evaluated using keystroke savings, which measures the percentage of keystrokes that were saved when using word prediction to type testing data as compared to letter-by-letter entry. Keystroke savings is typically measured using a simulated user that selects the desired word as soon as it appears. This kind

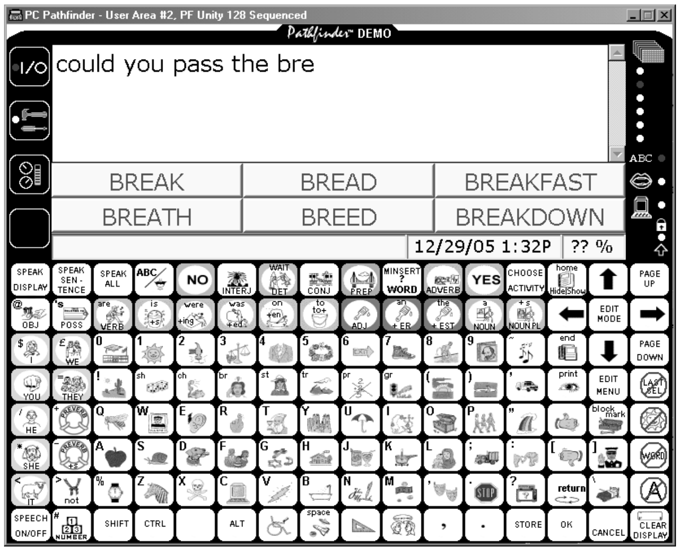


Figure 4: Pretnke Romich Company's PC Pathfinder Demo, a digital copy of the physical Pathfinder device.

of evaluation takes into account the quality of the predictions generated by the language model, but also user interface settings such as the number of words in the prediction window. Although the ultimate goal of word prediction is to enhance communication rate, this is very difficult due to a myriad of issues. Improvements in keystroke savings have been shown to have a positive affect on communication rate [76]. Keystroke savings will be discussed in detail in Section 2.1.

1.2 Guide to remaining sections

Our proposal is arranged as follows. Section 2 will apply a baseline trigram model to word prediction and demonstrate our evaluation methodologies. The results of evaluating a trigram model will help motivate the adaptive language models of later sections. Section 3 will present our solution to topic adaptation, showing that topic modeling can improve word prediction, even when tested on very different texts. We will present many of the lurking problems in topic modeling along with our solutions. Section 4 will outline our plans in applying our experience with topic modeling to style adaptation as well as cache-based adaptation. Section 5 will describe our plans for combining our topic, style, and cache adaptations to create a single overall model that simultaneously adapts to the three types of information. We will also present a way of evaluating the success of the combination approach independently of the individual models. Finally, Section 6 will enumerate our existing and expected contributions.

2 Motivation for Adaptation: Corpus Studies

The lack of appropriate corpora complicates the training of ngram models. However, it also complicates testing — without corpora from the target population, how will we know if our techniques will benefit AAC users? We will first address the problem of computing keystroke savings in Section 2.1, presenting many of the hidden issues with keystroke savings along with our decisions in applying keystroke savings. We will address corpus issues in Section 2.2, evaluating word prediction on a number of different corpora as well as varying the relationship between training and testing texts to allow for better interpretation. Section 2.2 will also serve as an example of typical values for keystroke savings.

2.1 Keystroke savings

The primary evaluation of word prediction systems is keystroke savings, the percentage of keystrokes that were not typed under word prediction as compared to letter-by-letter typing. The standard form of keystroke savings (KS) is shown below:

$$KS = \frac{keystrokes_{\text{normal}} - keystrokes_{\text{with prediction}}}{keystrokes_{\text{normal}}} * 100\% \tag{1}$$

However, the above equation leaves two questions unanswered: What is a keystroke? How should the number of keystrokes under each entry method be computed? We will address the first question of the keystroke definition in detail and then the second question briefly.

The definition of a keystroke for evaluation includes two sub-parts: what actions are keystrokes and which kinds of keystrokes count towards keystroke savings? For the first subproblem, most researchers take characters to be keystrokes. The main exception to this is uppercase letters.[2] The problem of measuring a keystroke is essentially reduced to the question of whether an uppercase letter is counted as one or two keystrokes. We follow the current trend that an uppercase letter is counted as a single keystroke. We expect this to affect the results very little, as uppercase words are much less common than lowercase words, especially for conversational language (see Trnka and McCoy [75]).

The keystrokes used for evaluation may differ from the keystrokes originally used to enter the text, because we evaluate using non-AAC texts. Firstly, many corpora have punctuation marks, but an AAC user in a conversational setting is unlikely to use punctuation due to the high cost of each key press. Therefore, we remove punctuation on the outside of words, such as commas and periods, but leave word-internal punctuation intact, such as dashes. Researchers differ on including spaces in evaluation or not. Some researchers only measure keystrokes used to produce words, not keystrokes used to separate words. Researchers that include spaces in their keystroke savings implement their prediction window to automatically add a space after selecting a prediction. For the most part, so long as systems being compared either all simulate spaces or all ignore spaces, the keystroke savings measure can still be used for comparison. However, there does not seem to be a predominant setting and most publications ignore this issue. Because we feel that spaces give results that are more realistic of the actual user's experience, we count spaces as keystrokes. A similar problem occurs for a newline or "speak key", which the user would press after completing an utterance or sentence. We are unaware of any prior publications that acknowledge this issue or even state whether such a keystroke is included in evaluation or not. However, we feel that the simulation of a speak key will produce an evaluation metric that is closer to the actual user's experience, therefore we include a speak key in our evaluation.

The second major problem in the above equation is computing how many keystrokes a user would take under each condition (letter-by-letter entry and word prediction). The common trend in research is to simulate a "perfect" user that will never make typing mistakes and will select a word from the predictions as soon as it appears. From the perspective of the system, the simulated user is an optimization algorithm which enters text using the minimum number of keystrokes. However, implementation of an optimal way of using the predictive interface is not always straightforward. For example, consider the predictive interface in Microsoft Word: a single word is predicted is offered as an inline completion. If the prediction is selected, the user may backspace and edit the word. However, this freedom makes finding the minimum sequence of keys more difficult — now the user may select a prediction with the incorrect suffix and correct the suffix as the optimal action, such as wanting to type "record", but having "records" as the prediction after typing only "re". We felt that a more intuitive interface would allow a user to undo the prediction selection by pressing backspace, so we implemented that functionality in our user study [76]. Because we decided that a user interface without this optimization problem was best, we simulated the user under the assumption that the perfect user would not use backspace. In our system, if the desired word does not appear in the predictions (with the desired suffix), the user would need to continue typing letter-by-letter. In addition to the problem of backspacing, any future research in multi-word prediction will face a similar problem, analogous to the garden-path problem in parsing, where a greedy approach does not always

[2] A secondary exception is accented characters, but these are very uncommon in English.

give the optimal result. These two examples (backspace-editing and multi-word prediction) illustrate some of the hidden complication involved in applying keystroke savings to evaluation.

In summary, we simulate the user typing all keys that produce a word, all spaces, and all newlines. We do not simulate punctuation and we do not allow the simulated user to backtrack-edit a prediction that can be easily converted into the desired word.

In addition to some of the complications in computing keystroke savings, the interpretation of keystroke savings merits discussion. Firstly, there seems to be a limit to the keystroke savings of word prediction — researchers have noted that it seems extremely difficult to improve keystroke savings beyond a certain point. Copestake [20] discussed the entropy of English to suggest that 50–60% keystroke savings on free text with a reasonable window size can be expected in practice, but that much more savings would be unreasonable to expect. Lesher et al. [54] replaced the language model in a word prediction system with a human to try and estimate the limit of keystroke savings. They found that humans could achieve 59% keystroke savings with access to their advanced language model and that their advanced language model alone achieved 54% keystroke savings. They noted that one subject achieved nearly 70% keystroke savings on one particular text, and concluded that further improvements on current methods are possible.

We investigated the problem of the limitations of keystroke savings first from a theoretical perspective, noting that keystroke savings can never reach 100%, unlike other evaluations. 100% keystroke savings would mean that the user didn't press any keys at all and the system divined the entire text they intended. Clearly this is impossible; some minimal amount of user input is required to produce a message. The minimum amount of input required corresponds to a perfect word prediction system — one that predicts every word as soon as possible. In a word *completion* system, the predictions are delayed until after one character is entered. In such a system, the minimum amount of input using a perfect language model is two keystrokes per word — one for the first letter and one to select the prediction. The system would also require one keystroke per sentence. In a word *prediction* system, the predictions are made immediately, so the minimal input for a perfect system is one keystroke per word (to select the prediction) and one keystroke per sentence. We added the ability to measure this maximum to our simulation software, which we call the *theoretical keystroke savings limit*. To study this, we evaluated a baseline trigram model under two conditions on the Switchboard corpus. The simulation software was modified to output the theoretical limit in addition to actual keystroke savings at various window sizes. To demonstrate the effect of the theoretical keystroke savings limit on actual savings, we evaluated the trigram model under conditions with two different limits — word prediction and word completion. The evaluation of the trigram model using word *completion* is shown in Figure 5. The actual keystroke savings is graphed by window size in reference to the theoretical limit. Keystroke savings increases with window size, but with diminishing returns (this is the effect of placing the most probable words first). One of the problems with word completion is that the theoretical limit is so close to actual performance — around 58.5% keystroke savings compared to 50.8% keystroke savings with five predictions. At only five predictions, the system has already realized 87% of the possible keystroke savings. Under these circumstances, it would take a drastic change in the language model to impact keystroke savings. The problem of the theoretical limit only worsens at larger window sizes.

We repeated this analysis for the word *prediction* condition, shown in Figure 6 alongside word completion. Word prediction is much higher than completion, both theoretically (the limit) and in actual keystroke savings. Word prediction offers much more headroom in terms of improvements in keystroke savings, so we focus on word prediction. However, it is important to consider the theoretical keystroke savings limit when analyzing lower keystroke savings than expected on a new corpus. It may be the case that some corpora generally favor shorter words, which would lead to a lower theoretical keystroke savings limit and most likely lower actual keystroke savings.

This analysis demonstrates a limit to keystroke savings, but this limit is slightly different than Copestake [20] and Lesher et al. [54] seek to describe — beyond the limitations of the user interface itself, there seems to be a limitation due to the predictability of English. Ideally, we would like to be able to recognize this limit by applying our work in the theoretical limit to more practical measures. One way in which we can derive a more practical limit is to simulate word prediction using a perfect model of all words that occur in the training data. In other words,

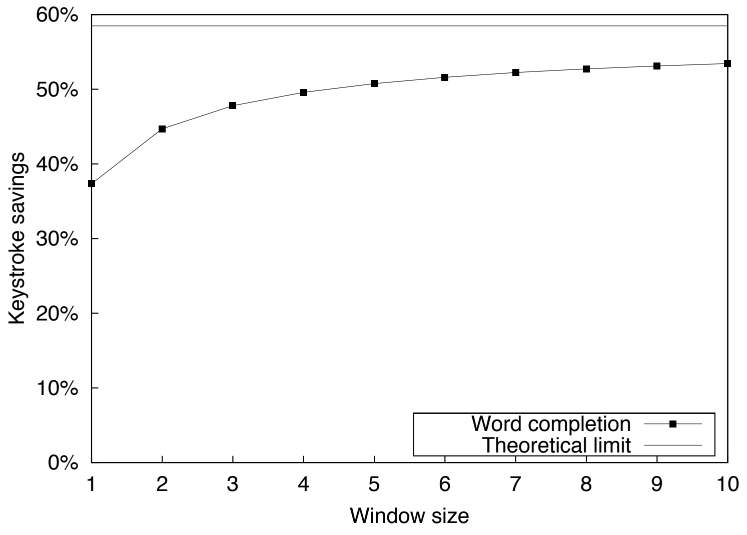


Figure 5: Keystroke savings vs. window size with word completion. The theoretical limit is shown above the curve.

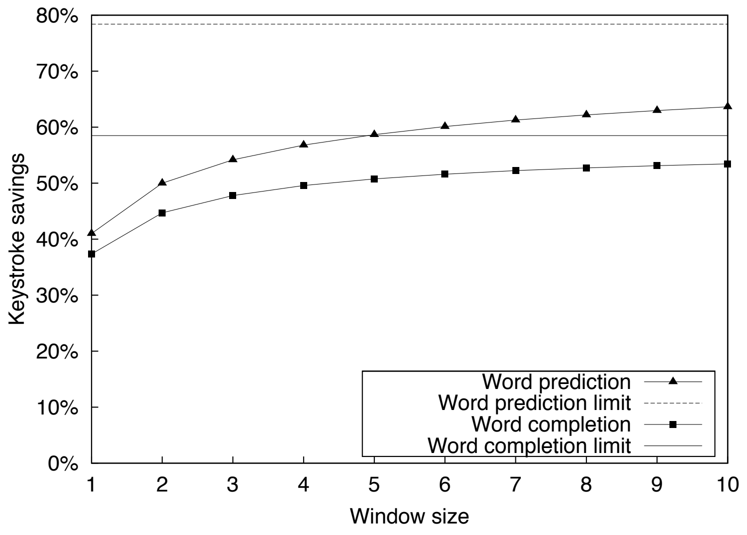


Figure 6: Keystroke savings vs. window size with word completion and word prediction. The theoretical limit of word prediction is much higher than word completion. The actual performance of word prediction is also much higher.

the language model will predict the correct word instantly so long as it occurs in the training corpus. Words that never occurred in training require letter-by-letter entry. We call this measure *practical maximum keystroke savings* and use it to try and evaluate whether the difference between training and testing vocabulary is significant. The difference between the practical and theoretical limit can be considered graphically also — due to vocabulary limitations, actual keystroke savings will never reach the theoretical limit as the window size grows. However, actual keystroke savings will eventually reach the practical limit when the window size is close to the vocabulary size.

Figure 7 shows the actual keystroke savings of word prediction along with the theoretical and practical limits. In the case of tests using Switchboard for training and testing, the practical limit is very close to the theoretical limit. Therefore, the remaining gap between the practical limit and actual performance must be due to other differences between testing and training data, limitations of the language model, and limitations of the predictability of English.

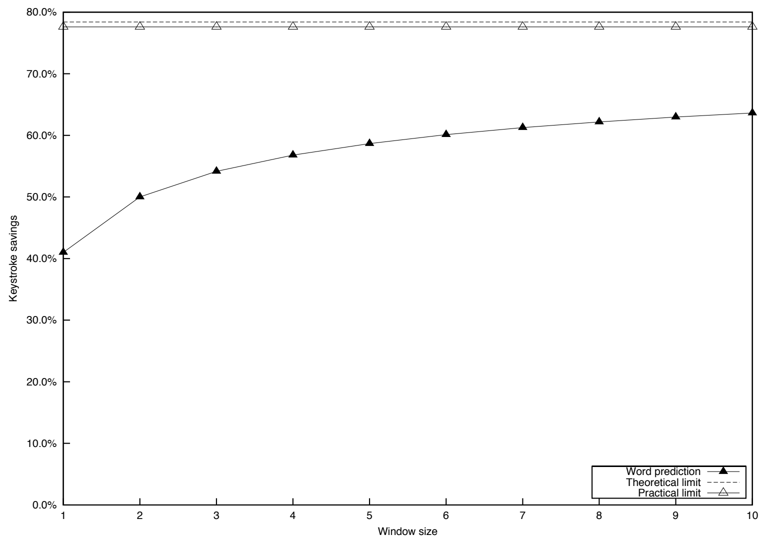


Figure 7: Keystroke savings vs. window size with word prediction. The practical limit of keystroke savings is shown in addition to the theoretical limit and actual performance.

The practical and theoretical limits of keystroke savings give some insight into research in word prediction. If actual performance is very close to the theoretical limit, then improvements in keystroke savings are inhibited by the user interface, and improvements to the interface should be the most beneficial. For example, if actual keystroke savings were close to the theoretical limit with word prediction, one way to increase the theoretical limit would be to investigate multi-word prediction. If actual performance is very close to the practical limit, but there is a wide separation between the practical and theoretical limits, then vocabulary issues must be addressed. Research in cache-based modeling is one way to address this. Better generalization of training data may be another way to address this. In the case above with word prediction, the actual keystroke savings is far from both limits, so performance is limited by either the quality of the language model or the predictability of English. We feel that performance is still primarily limited by the quality of the language model, especially when considering the miniscule amount of information that a trigram model uses in word prediction.

2.2 Corpus issues

The results of evaluation are heavily dependent on the characteristics of the texts used for training and testing. For example, one of the often noted factors in evaluation is the effect of the number of words used to train a language model [55]. Language models built from larger corpora tend to perform much better, particularly on words that are infrequent. However, another determinant of performance is how well the language training data reflects the actual language the system is to be run on. Statistical systems tend to perform poorly when they are applied to language very different from the training texts [20, 83, 67]. Thus, important questions include: What text should be used for training and testing a word prediction system? We address this question in this section as well as a publication [75].

Traditionally, most researchers have performed what we call *in-domain evaluation* [55, 52, 28, 26, 57] — a corpus of text is split into training and testing sections, where the training section is used to build the ngram language model and the testing section is used to evaluate the quality of the predictions.[3] Splitting a corpus into training and testing sets is the most common means of evaluation in NLP because it gives a "fair" evaluation of each method of language model development. This allows two different methods to be compared by holding constant the training and testing data so that any differences in the methods can be attributed to language model development (such as [58, 7, 70, 27]). However, this evaluation gives only a vague indication of end-user benefit from various techniques.

Since the actual use of the system might be with language data that is quite different from the training corpus, some researchers have performed *out-of-domain evaluation* [80, 11, 59] — evaluating their predictions on text not from the training corpus. This approach gives a more reasonable estimate of real-world performance, especially in situations where the actual user text is likely to differ from the training corpus. The same approach is sometimes used in the more general field of language modeling to validate that the performance improvement on in-domain data still applies when tested out-of-domain (e.g., [7]). However, out-of-domain testing is not always a fair evaluation to verify that new techniques are working. Poor results in the out-of-domain testing corpus may be a result of differences in language use between the training and the testing corpus and may not carry over to the actual domain of use. For instance, trying to apply a method like topic modeling to a testing corpus containing mostly general conversations is unlikely to show much difference between different variations in topic modeling, while the real-world difference may be highly significant.

Some researchers have also performed *mixed-domain evaluation* where some of the training data is from the same corpus as the testing data, but much of it is from other corpora. The incorporation of recency information into the prediction method is an example of this [83] — the dynamically updated collection of user text is in-domain and the baseline model is out-of-domain. Mixed domain evaluation has also been used in a part-of-speech (POS) framework, where the sequences of POS tags are trained using a very small in-domain corpus and the word-based probabilities are trained using a larger out-of-domain corpus [20].

Realistically, the question of in-domain vs. out-of-domain testing is a spectrum — training on phone calls between family and testing on face-to-face communication between family is closer to in-domain testing than training on phone calls between family and testing on newspaper articles. There are various dimensions along which corpora can be similar or different, but we can group the dimensions roughly into topic and style, where topic covers the content of communication and style describes variations such as formality, speech repairs, opinionated vs. objective communication, common vs. uncommon word choices, etc.

In addition to the problem of selecting similar or dissimilar text for training and testing, developing a system for AAC users is hampered by the lack of substantial corpora of AAC text. We feel that non-AAC conversations will use longer sentences with many more speech repairs than AAC text. Therefore, evaluations on non-AAC text may not be representative of AAC users. The three ways in which we address this problem are 1) to select corpora that are similar to AAC text, 2) to transform text to be more AAC-like when possible, and 3) to construct a small corpus of AAC text.

[3] Generally cross-validation is performed to lessen the sensitivity of evaluating on a particular training-testing split.

Section 2.2.1 will describe the corpora we have chosen to evaluate word prediction as well as the cleanup we have performed to bring them closer to what we think an AAC user would type. Section 2.2.2 will present a case example of evaluation using the baseline trigram model and analysis with domain variations.

2.2.1 List of corpora

AAC devices are used for a wide variety of communication needs — everything from spontaneous conversation to preplanned speeches to homework assignments and technical articles. For this reason, we feel that a variety of texts should be used to evaluate word prediction. However, obtaining a large corpus of AAC text can be difficult. To address this issue, we have assembled a variety of corpora.

Because spoken conversation is arguably the most common use for an AAC device, we first assembled several corpora of spoken English and performed cleanup processing to remove speech repairs, bringing the text closer to what an AAC user might say. Because AAC devices can be used for writing as well, we also assembled a small collection of emails from AAC users and included a corpus of written text to get a rough idea of any differences in word prediction in written vs. spoken text. Each of the corpora came with their own formatting conventions: most were in all lower case (except for proper names) but some (primarily the written corpora) used a capital letter to start each new sentence. Some omitted punctuation while others included commas and other punctuation symbols within a sentence. To best reflect the primary usage of AAC devices in speech, we reformatted the corpora to reflect a standard style. Therefore, "I" and contractions such as "I'll" were capitalized in all corpora. The first word of each sentence was converted to lowercase unless it was a known named entity, which would remain capitalized. Also, punctuation between words in a sentence was removed. We feel that these changes make the collection of corpora more AAC-like and facilitate a more fair evaluation of word prediction. The overall collection is roughly half spoken and half written, shown in Figure 1.

Corpus	**Medium**	**Word count**
AAC Email	email	27,710
Callhome	spoken	48,407
Charlotte	spoken	187,587
SBCSAE	spoken	237,191
Micase	spoken	545,411
Switchboard	spoken	2,883,774
Total spoken	spoken	3,902,380
Slate	written	4,178,543

Table 1: Word counts for each corpus (see corpus-specific sections below for more details)

2.2.1.1 Conversational Speech Transcriptions

Ideally, we would like to evaluate word prediction with conversational AAC text, but thus far, such a corpus has been unavailable. Instead, we will evaluate word prediction on several conversational speech texts. However, there are differences between these texts and what we would expect from an AAC corpus. In particular, spoken text contains many speech repairs — false starts, word repetition, uhs, ums, etc. Therefore, we removed speech repairs in an effort to bring the corpora closer to what an AAC user would type.

Speech repair removal Transcribed conversational text is characterized by frequent speech repairs. Many speech repairs are the result of "getting ahead of oneself", such as suggested by Shriberg [71]. However, AAC communication rate is more limited by the speed of producing words rather than the speed of planning a message. For this reason, speech repairs were removed when they could be easily identified. We follow the work of Hindle

[36] in processing simple speech repairs such as backchannels (e.g., uh, um), repetitions, and limited cases of word replacements.

We used pauses, abandonment of words, and backchannels as candidates for being an editing signal, depending on what was annotated in each corpus. For example, one corpus (Switchboard) clearly marked words as abandoned, whereas in other corpora we relied on commas and multiple periods to signal pauses. Each sentence would have candidate editing signals identified and then lexical pattern matching was performed on the words to the left and right of the potential editing mark. In the case of abandoned words, exact matching wasn't required. However, some speech repairs (e.g., "I I would ...") were not signaled. Therefore, we created special processing for single-word repetitions: repeated lowercase words were considered speech repairs unless they appeared in an exceptions list. Repeated uppercase words were considered *legitimate repetitions* unless they appeared in an exceptions list, which primarily contained derivations of "I". Any backchannels that remained after speech repair removal were filtered out. The resulting "cleaned" text was far easier for the authors to read and is much closer to what we think an AAC user would have said.

We employed several different corpora of spoken conversation:

Switchboard The Switchboard corpus is a collection of 2,438 English phone conversations recorded by Texas Instruments using a variety of speakers and topics [74]. Participants indicated which of the predefined topics they were comfortable discussing and the experimental software connected two subjects to speak about a particular topic, given by a prompt such as "Find out what kind of fishing the other caller enjoys..." After the speech repair cleanup, there are roughly 2.9 million words in Switchboard — more than any other corpus of speech that we used. Although the task-focused nature of Switchboard makes it slightly unrealistic of unprompted day-to-day conversations, we feel that the large size of Switchboard outweighs any small dissimilarities.

SBCSAE The Santa Barbara Corpus of Spoken American English (SBCSAE) [68] is a collection of 60 recorded conversations, which are predominantly face-to-face communications and have been collected to sample a wide variety of speakers. As an example of the variety found in SBCSAE, it contains a social conversation held over lunch, a conversation on a ranch, and a church sermon. Although SBCSAE spans a wide variety of topics, it contains a mere 237,191 words — roughly 8% of the size of Switchboard. However, the natural nature of the text in SBCSAE is a step closer to the conversational communication of AAC devices.

Micase The Michigan Corpus of Spoken Academic English (Micase) is a collection of university-setting spoken English. Several example conversations are advisor-advisee discussions or moderated class discussions. We obtained a portion of the Micase corpus though the second release of the American National Corpus (ANC) [2]. Special processing was added for parentheticals and quotations to focus the ngram model on the proper conditioning information. The Micase data we used contained 545,411 words across 50 conversations. Although the text isn't representative of most day-to-day speech, it should be representative of word prediction performance for other highly specialized conversations, such as speech in the workplace.

Callhome The Callhome corpus is represented in part in the ANC corpus, and contains 24 telephone conversations between friends and family. This free-form conversation is very representative of day-to-day communication and is very appropriate to approximate AAC user text. Callhome contains a mere 48,407 words, but like SBCSAE, although the text is relatively small, it is a valuable approximation of day-to-day AAC user conversation.

Charlotte The Charlotte Narrative and Conversation Collection (Charlotte) is a collection of 93 narratives, conversations, and interviews centered around an area in North Carolina, USA, available as part of the ANC corpus. Charlotte contains 187,597 words, about 80% of the size of SBCSAE. This corpus is very similar in its conversational nature to Callhome and SBCSAE, and is therefore useful despite its small size.

2.2.1.2 AAC Email Corpus

AAC user text has been difficult to obtain, but one resource we found was a publicly available AAC user mailing list archive. We surveyed emails from this archive and collected emails from two posters who were known to be AAC users. The resulting corpus contains 117 emails and 27,710 words. Like several other corpora, this data is useful despite its small size because it's a test directly applicable to the target, AAC users. Email processing presented new challenges for cleanup processing. Signature text was removed, as an email user only types the text once, not for each email sent. Quoted emails in replies were also removed. In addition, parentheticals and quotations were extracted like with Micase.

2.2.1.3 Slate Magazine

Slate Magazine is an online publication covering a wide range of topics, similar to a newspaper. The ANC project contains 4,531 articles from Slate published in a span of 4 years. At 4,178,543 words, Slate is the largest corpus in this study. We feel that it serves as an approximation of one kind of written AAC text as well as a general-purpose corpus of English. The cleanup processing for Slate was similar to the AAC Email Corpus with the exception of small adjustments to make articles in Slate more natural (e.g., removing article titles).

2.2.2 Example domain-varied evaluation: Trigram baseline across domains

The overall goal of this part of our research is to evaluate how users will benefit from word prediction. In practice, an AAC device is using a language model from a different topic or style to predict words in their real conversation. We use three tests to evaluate this for each corpus: in-domain, out-of-domain, and mixed-domain training. In-domain training uses the same corpus for both the training and testing sets. Out-of-domain training uses the training sets of all corpora except the corpus used for testing. Mixed-domain training uses the training sets of all corpora and evaluates on the testing set of each corpus. In-domain training is the most common means of evaluating word prediction (e.g., [55, 52, 28, 26, 57]), so it forms a baseline to which other training sets can be compared. In-domain performance is determined primarily by the size of the corpus and the intrinsic complexity of the corpus (where this includes the corpus' self-similarity). Out-of-domain training shows the expected degradation (or improvement) of performance resulting from using word prediction in a realistic scenario. The difference between in-domain and out-of-domain performance should be proportional to the difference in training data size and the differences in language between in-domain and out-of-domain text. Mixed-domain training approximates what a high-performance system might do: incorporate user text back into training in addition to a large out-of-domain data set. Mixed-domain performance should be scrutinized with respect to *both* out-of-domain and in-domain performance, as it subsumes both training sets.

We evaluated the baseline trigram model using in-domain evaluation first, shown in Table 2. The corpora are ordered from smallest (AAC Email) to largest (Slate). The keystroke savings under in-domain testing roughly correlates with the size of the corpus — the largest three corpora also have the highest keystroke savings, especially Slate and Switchboard. The three largest corpora also have the highest self-similarity using a cross-validation OOV test, presented in [75]. One notable exception to the trend of corpus size is the AAC email corpus, which shows higher keystroke savings than the much larger Callhome, Charlotte, and SBCSAE. The self-similarity of the corpus may be responsible — the emails were sampled from only two AAC users. In contrast, we think that there is less data per speaker in other corpora, despite the larger amount of text overall. Also, Switchboard shows higher keystroke savings than the much larger Slate corpus. As with AAC Emails, this may be due to the homogeneity of Switchboard, which is comprised of only 70 topics. In addition to the much more restricted number of topics in Switchboard, we balanced the cross-validation sets to have comparable topic distributions. This reduces the chances of encountering a word in testing that did not occur in any training texts.[4]

We also evaluated the baseline trigram model using out-of-domain training in Table 3. The results of in-domain training are included for comparison and the highest keystroke savings for each corpus is bolded.

The much larger amount of training data contributed to higher keystroke savings on several corpora — the training

[4]Topic balancing was performed to provide a more thorough test of our topic modeling approach in Section 3

Corpus	**In-domain training**
AAC Email	48.92%
Callhome	43.76%
Charlotte	48.30%
SBCSAE	42.30%
Micase	49.00%
Switchboard	60.35%
Slate	53.13%

Table 2: Keystroke savings with in-domain evaluation.

Corpus	**In-domain**	**Out-of-domain**
AAC Email	**48.92%**	47.89%
Callhome	43.76%	**52.95%**
Charlotte	48.30%	**52.44%**
SBCSAE	42.30%	**46.97%**
Micase	49.00%	**49.62%**
Switchboard	**60.35%**	53.88%
Slate	**53.13%**	40.73%

Table 3: Keystroke savings of in-domain vs. out-of-domain training. The maximum keystroke savings for each row is shown in bold.

data size increase ranged from 166 times more data for Callhome (for 9.19% more savings) to 13 times more for Micase (for 0.62% more savings). Interestingly, a notable exception to this rule is the AAC Email Corpus which performs about a percent worse using a much larger amount of out-of-domain data (almost 300 times the in-domain training data). Part of the reason for this is that the email genre is very different from the other corpora and also the topic of AAC Emails may be somewhat different than the other corpora, as with the effects of self-similarity for in-domain training. In contrast to the smaller corpora, the larger corpora showed better keystroke savings using in-domain training. The results on Slate are somewhat unsurprising, as there is less out-of-domain training data compared to in-domain. The results on Switchboard are noteworthy — the out-of-domain training is slightly larger than the in-domain training size. In this case, the lower keystroke savings of out-of-domain can be attributed to the very difference style and vocabulary of the other corpora, which is dominated by the formal written style of Slate.

We also evaluated an estimation of what a user-adaptive model might do — we used both the in-domain and out-of-domain training texts for mixed-domain training, as shown in the last column of Table 4. As with the out-of-domain test, the highest keystroke savings for each corpus is shown bolded.

Mixed-domain training shows that even a simplistic mix of a small amount of in-domain data with a large amount of out-of-domain data can increase keystroke savings. The most notable increase here was found in the AAC corpus which improved (3.3% – 4.3%) over both in-domain and out-of-domain training. Callhome, Charlotte, SBCSAE, and Micase also save more keystrokes using a mix of training data over either training set alone. The larger corpora, Switchboard and Slate, show a performance loss with mixed training over the in-domain models — the out-of-domain data “distracts” the language model from the in-domain data. This distraction is a similar trend for all corpora with in-domain training, however, the in-domain trigram models for Switchboard and Slate were already fairly reliable, whereas the in-domain trigram models were much less reliable for the much smaller corpora. The performance improvement on the AAC Email Corpus in particular is astonishing considering it contributes such a small fraction of the probability mass of the learned model.

This analysis has demonstrated a way of comparing word prediction results across domains. If we train a language

	Training domain		
Corpus	**In**	**Out**	**Mixed**
AAC Email	48.92%	47.89%	**52.18%**
Callhome	43.76%	52.95%	**53.14%**
Charlotte	48.30%	52.44%	**53.50%**
SBCSAE	42.30%	46.97%	**47.78%**
Micase	49.00%	49.62%	**51.46%**
Switchboard	**60.35%**	53.88%	59.80%
Slate	**53.13%**	40.73%	53.05%

Table 4: Keystroke savings of in-domain vs. out-of-domain vs. mixed-domain training. The maximum keystroke savings for each row is shown in bold.

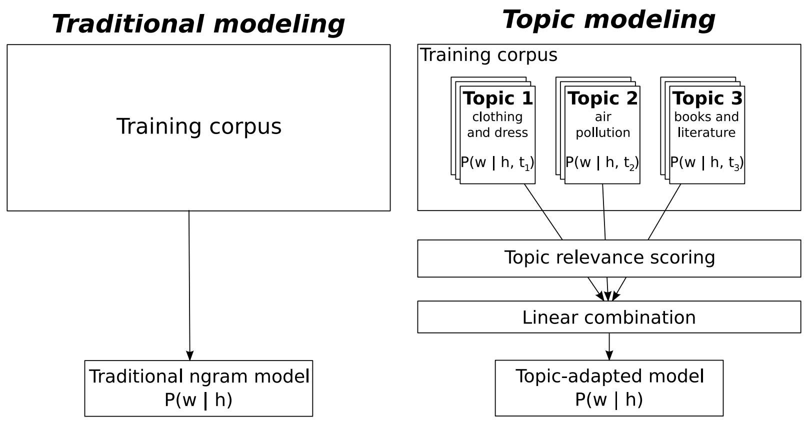
model on Switchboard and test on both Switchboard and Slate (the out-of-domain test on Slate is close to this), we would expect to see lower keystroke savings on Slate. The in-domain test on Slate seems to indicate that word prediction on Slate is difficult, perhaps due to the diversity and formality of Slate.

We have designed this testing methodology for two reasons: Firstly, we wanted to devise an evaluation for AAC devices that was as realistic as possible in light of the lack of AAC corpora. Secondly, we wanted to have a suite of corpora to use for evaluation, partly to reflect the varying uses for word prediction and partly to reduce overfitting our research any one particular corpus. Finally, we developed the domain-varied evaluation to have a better understanding of real-world performance, where the text used in testing is different from the text used in training. Many of the results of this methodology are presented in Trnka et al. [75].

2.3 Motivation for Adaptation

The investigation into the limits of keystroke savings demonstrates that any further improvement in keystroke savings is likely to come from language modeling improvements, not a reduction in the minimum number of keystrokes per utterance or an increase in the vocabulary size. The evaluation of practical maximum keystroke savings in particular suggests that the language model has the data to predict words quite well, however, a baseline trigram model is not an ideal model of the training data.

The results of the corpus study shed further light on the problem. Training data size and the similarity of training and testing data have a strong effect on keystroke savings. Keystroke savings increases with more training data, but not at the expense of trading a small amount of very similar data for a large amount of very dissimilar data. Regardless of how much training data we use, additional data beyond the most relevant texts may not improve keystroke savings, or worse, may degrade keystroke savings by washing out the most relevant data. Rather than manually selecting appropriate training data and balancing the size of the training data with the similarity to the testing domain, we have developed (and will further develop) methods that automatically identify the most similar portions of the training data and focus training on those parts. At the same time, our methods will allow all training data to have an effect, which allows us to focus training on the most topically and stylistically relevant portions without reducing the effective size of the data. In addition to these general lessons from the corpus study, the AAC Email results provide strong motivation for cache modeling. The mixed-domain case combines the AAC Email training data with nearly 300 times as much out-of-domain data, yet the naïve combination of the two training data sets vastly outperforms either training set alone. This suggests that even if we are unable to find data for each domain, a well-designed cache model should be able to substantially reduce the problem of lack of appropriate training data, and potentially even outperform a model trained on in-domain data if only limited data is available.



(a) Traditional ngram model (b) Topic model

Figure 8: Topic modeling compared to traditional ngram modeling at the high level

3 A Study in Adaptation: Topic Modeling

Topic-adapted language models have been used to improve both word prediction [57, 52, 77, 78] as well as speech recognition [7, 58, 70, 27, 49, 65]. Our method of topic adaptation utilizes corpora that are labeled by topic[5]. Topic models focus on the topics in training that are most relevant to the current context, in contrast with traditional ngram models, which treat all training data equally, illustrated at a high level in Figure 8.

We will present our research in topic modeling in Sections 3.1, 3.2, and 3.3. Section 3.1 will investigate the type of language models associated with each topic and investigate the application of backoff to topic modeling. Section 3.2 will describe how we identify which topics are most relevant to the current discourse. Section 3.3 will investigate variations in topic definitions — for example, how the model is affected if we use documents as topics rather than human-annotated topics. After our own findings have been presented, we will compare and contrast them to other researchers in Section 3.3.5.

3.1 Overall model

We use the Markov model framework to build topic models, where the overall model is a linear combination of individual topic models, weighted by likelihood. In this manner, even if the likelihoods of a few topics are poor, all topics contribute, so the effect of mistakes in weighting the topics is minimized. This addresses the problem noted by Lesher and Rinkus [52] that selecting the most appropriate topic for training is worse than a traditional ngram model trained on all training data, due to problems of data sparseness. The high-level topic model takes the form:

$$P_{topic}(w \mid h) = \sum_{t \in topics} P(t \mid h) * P(w \mid h, t) \tag{2}$$

where w is the word being predicted/estimated, h represents all of the document seen so far, and t represents a single topic.

The remainder of this section will focus on the estimation of $P(w \mid h, t)$ (the ngram model for topic t) and how the overall model is built around it. The typical approximation of h is to make a Markov assumption that only the previous couple words affect the current word, for example, simplifying $P(w \mid h, t)$ to $P(w \mid w_{-1}, w_{-2}, t)$ in the case of the previous two words. We also have an additional way to reduce the number of parameters — we can estimate local constraints $P(w \mid w_{-1}, w_{-2})$ and global constraints $P(w \mid t)$ separately and combine

[5]Though we experiment with relaxing this requirement in Section 3.3.

them afterwards (discussed in Section 3.1.1). Section 3.1.1 will discuss a range of options in implementing the ngram model for each topic. The remaining sections will address several problems in applying traditional ngram techniques such as backoff and smoothing to topic modeling.

3.1.1 Pure and hybrid topic modeling

We have explored two options in implementing $P(w \mid h, t)$ — either this model can be a full-fledged ngram model or it can be an impoverished model which is later combined with a full-fledged baseline model. In the first approach, which we call *pure topic modeling*,[6] we investigated the use of a trigram model for each topic in [77] — $P(w \mid w_{-1}, w_{-2}, t)$. Note however, that research using 4-grams with pure topic modeling would compute a 4-gram model for each topic, and likewise for any other ngram model. This approach follows researchers such as Seymore and Rosenfeld [70], Florian and Yarowsky [27], and Mahajan et al. [58].

In the second approach, which we call *hybrid topic modeling*,[7] a simplistic language model is computed for each topic. In previous work [77], we measured a unigram model for each topic. The topic modeling approach in this case creates a topic-adapted unigram model, similar to the LSA-adapted model of Bellegarda [7, 5, 6]. Because unigrams are generally poor models of language, we combine the topic-adapted unigram model with a baseline ngram model trained on all data.

$$P_{hybrid}(w \mid h) = P_{baseline}(w \mid h) * \left(\sum_{t \in topics} P(t \mid h) * P(w \mid t) \right)^{\alpha} \tag{3}$$

where $P_{baseline}(w \mid h)$ is a non-adaptive baseline model from all training data and $P(w \mid t)$ is a unigram model trained on topic t. Unlike Bellegarda [7], we found it necessary to use the weighting factor α to improve keystroke savings. In an application where exact probabilities are necessary, the equation would be normalized by taking the $1 + \alpha$th root, but since we use the results only for ranking, true probabilities are not necessary.

In fact, these two models are at opposite ends of a spectrum. At one end of the spectrum, each topic produces a full-fledged ngram model and at the other end, topic affects the probability of individual words only, and is afterwards combined with a full-fledged model. We can think of both approaches as fitting into a general framework $P_{dynamic}(w \mid h_1, t) * P_{static}(w \mid h_2)$, where $P_{dynamic}$ is the topic-adapted component and P_{static} is the baseline component. In this representation, h_2 must contain at least as many words of context as h_1. Pure topic modeling is similar to approximating h_1 and h_2 using the same Markov assumption and omitting the static model. At the hybrid end of the spectrum, h_1 is empty and the contextual dependence of the model is fully placed in the static model. In the middle of the spectrum, the two approximations of the history are not identical ($h_1 \neq h_2$) and both have at least one word of context. For example, a topic-adapted bigram model could be combined with a static 4-gram model.

We evaluated and compared pure and hybrid topic modeling in [77]. In that paper, pure topic modeling is called Method A and hybrid topic modeling is called Method B. Also, due to older hardware requirements, we implemented pure topic modeling with bigrams rather than trigrams; a trigram model was used for the static baseline in hybrid topic modeling. Aside from the ngram order, all other implementation details were identical between the pure model and the unigram topic component of the hybrid model. We applied hand-tuning in the range $[0.05, 2]$ to determine a suitable weight α for the hybrid model, and found that $\alpha = 0.05$ was the best setting for keystroke savings. This is unfortunate, as it means that the topic-adapted portion of the equation is weighted very little and therefore, topic adaptation is unlikely to affect the model as much.

We found that both approaches increased keystroke savings over the baseline trigram model, with pure topic modeling improving results by 0.8%–1.5% and hybrid topic modeling improving the results by 0.2–0.4% (shown in Table 5). Because hybrid modeling required only slightly more computational resources than the baseline model,

[6] In earlier work [77], this was called Method A.
[7] In earlier work [77], this model was called Method B.

Window	Bigrams	Trigrams	Pure topic (bigrams)	Hybrid topic (trigrams)
1	41.5%	42.3%	43.1% (+0.8%)	42.5% (+0.2%)
2	50.6%	51.1%	52.3% (+1.2%)	51.4% (+0.3%)
3	54.7%	55.1%	56.4% (+1.3%)	55.4% (+0.3%)
4	57.0%	57.3%	58.7% (+1.4%)	57.7% (+0.4%)
5	58.6%	58.8%	60.2% (+1.4%)	59.1% (+0.3%)
6	59.8%	60.0%	61.4% (+1.4%)	60.3% (+0.3%)
7	60.6%	60.8%	62.2% (+1.4%)	61.1% (+0.3%)
8	61.3%	61.5%	62.9% (+1.4%)	61.8% (+0.3%)
9	61.9%	62.0%	63.5% (+1.5%)	62.3% (+0.3%)
10	62.4%	62.5%	64.0% (+1.5%)	62.8% (+0.3%)

Table 5: Pure topic modeling with bigrams and hybrid topic modeling with trigrams compared to static trigram and bigram baselines. The improvement over the trigram baseline is shown in parentheses.

whereas pure modeling required much more time and memory, we concluded that hybrid topic modeling might be suitable for devices with limited resources when maximum keystroke savings was desired but not absolutely necessary, whereas pure topic modeling would be more applicable when absolute keystroke savings was more important.

There are a couple of interesting trends between pure and hybrid topic modeling. First, the pure topic model offered greater improvement at higher window sizes. This is due to the usage of bigrams in the pure topic model compared to trigrams in the baseline. For a similar trend, note the differences between the bigram and trigram baseline. The trigram-based predictions show a large difference from bigrams at smaller window sizes, but as the window size increases, the difference is substantially diminished. We feel that this trend is in part due to the sparseness of our data; the trigram model does not produce a whole 10 predictions because it is sparse, so in the event of large window sizes and infrequent trigram histories, many of the predictions in the trigram approach are generated using backoff to bigrams (backoff will be covered in the following section). The difference between bigrams and the bigram-based pure topic modeling fluctuates in the closer range 1.6% – 1.7%.

The second trend is that hybrid topic modeling offers much less benefit than pure topic modeling. We feel that part of the reason is that the overall model of the hybrid approach is only affected slightly by topic adaptations due to the weight of 0.05 on the topic-adapted unigram model. This weight was found by tuning to optimize keystroke savings, so a higher weight would have decreased performance, even though it would allow topic adaptations to have a greater effect. We feel that the main problem in the hybrid approach is that the context-based component (the static trigram model) and the topic-based component (the dynamic unigram model) are competing in a voting-like system to affect the overall model.

We attempted to fix this problem by forcing the predictions due to trigrams to occur higher in the prediction window than predictions from bigrams and bigrams higher than unigrams. Then we had the topic-adapted unigram re-rank the predictions within each block in conjunction with the probability from the static model. However, we found that the block model degraded keystroke savings both with and without the re-ranking. We concluded that the predictions due to the bigram model must have occurred higher in the list than trigram predictions in cases of a sparse trigram distribution, and gave thought to a model with trigrams and bigrams blocked together above unigrams using the same re-ranking within a block, but decided instead to focus on pure topic modeling.

Finally, we feel that the difference between pure and hybrid modeling is not only due to the low weight in hybrid modeling, but also due to pure topic modeling adapting a full bigram model to the topic of discourse. Intuitively, a topic model should adapt the predictions to include domain-specific vocabulary. However, vocabulary is not completely captured by a unigram model — compound words are common in specific domains. The hybrid model is able to adapt to the domain-specific single-word terms, but unable to adapt to compound words. Consider the case of predicting the second word in “word prediction”. In the case of a hybrid model that has correctly adapted to the word prediction topic, the word “prediction” will be boosted up in probability in any context. However, the context from *all* training data will be used as the base; if the word prediction topic was small in training and other topics using the word “word” were prevalent, then the context would have a poor starting point for topic unigrams to fine-tune probabilities. On the other hand, pure topic modeling adapts the *entire* ngram model to the topic of conversation within context. So a pure approach is far more likely to predict compound words in the topic. The ability of the pure model to adapt the effects of context as well as the vocabulary also allows it to handle more than just compound words, such as verb-object constraints when they are adjacent.

Pure topic modeling offers more benefit than hybrid topic modeling, even when used with bigrams. We feel that the added benefit offers more potential synergy with other language models, especially style modeling. In this approach, topically-related words will be filtered by style and stylistically appropriate words will be filtered by topic. In addition to the higher keystroke savings of pure topic modeling, the low weight on the topic-adapted unigrams of hybrid modeling means that any improvements in the topic unigram model will still be constrained by the weight; improvements to the topic model have less ability to affect the overall model due to the low weight. Therefore, our further research will focus on pure topic modeling.

3.1.2 Background in smoothing and backoff

Unfortunately, a single ngram model is often insufficient for language modeling. Especially with higher-order models, the sparseness of the model becomes problematic (i.e., a pure trigram model may only be able to generate 1–2 predictions in certain contexts). Therefore we need a way to take advantage of the strong contextual information from a higher-order ngram model while addressing data sparseness. We have chosen the backoff model paradigm to balance context and data sparseness, following Katz [43].

Backoff models are composed of multiple order ngram models. The highest order ngram model is checked first, and if the probability of the ngram is defined, it is returned. If not, the model checks the next highest model, and so on. In order to remain a probability distribution, each ngram model must be discounted and the held-out probability mass must be distributed to the lower order models in a recursive manner. This is shown below in the case of a trigram backoff model:

$$P'(w \mid w_{-1}, w_{-2}) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1},w_{-2}} * P_{smooth}(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1},w_{-2}} * P_{smooth}(w) & otherwise \end{cases} \tag{4}$$

where P_{smooth} is a discounted probability distribution over the specified history. The α values are the held-out probability mass such that:

$$\alpha = 1 - \sum_{w} P_{smooth}(w \mid h)$$

The discounting or smoothing algorithm is responsible for computing probabilities $P(w \mid h)$ from a frequency distribution $f(w \mid h)$, where the probabilities are adjusted from their MLE values to hold out probability mass for lower-order models. This component will be discussed further in Section 3.1.4.

The effect of backoff is that the most useful ngram models contribute the most to the backoff model, but lower-order models are consulted as necessary, iteratively pruning the conditioning information of the model. In this manner, the distribution iteratively becomes more reliable, but loses the conditioning information little-by-little. The effect is that the backoff language model tries to constrain words as much as possible, but still provides non-zero probabilities as often as possible.

While the history and development of smoothing and backoff are rooted in speech recognition, we are applying these techniques to word prediction. In word prediction, a language model with smoothing but not backoff is no more useful than an MLE model[8]; word prediction requires knowledge of the vocabulary, whereas smoothing by itself only fixes issues of zero probabilities for unknown words. If the probability of a word is zero, then the

[8]That is to say, discounting without some other technique is useless for word prediction.

language model is unable to predict the word, so whether the language model assigns a probability to an unknown word or not doesn't affect the prediction task at all. However, backoff is very useful in word prediction because it adds more words to predict. Because backoff is useful and because it requires smoothing to be performed first, smoothing is used in word prediction as a precursor to backoff. One final note about backoff in word prediction is that a final step of backoff to a uniform distribution is unnecessary, though using a dictionary as the final step of backoff rather than an even distribution based on the estimated vocabulary size is useful (as mentioned in Section A.1).

The remainder of this section describes the options we have in using smoothing and backoff — when to apply backoff and smoothing (and how to get the intended model to work well) and selecting a smoothing method that is appropriate for both a non-adaptive model as well as a topic model.

3.1.3 When to apply backoff and smoothing?

While significant existing research has been put into backoff and smoothing for standard ngram models, we are unaware of substantial research in applying these techniques on a linear combination of topic models. Backoff requires that smoothing is performed first, but the question remains: When in the combination model should backoff and smoothing be applied? Should we have a backoff model for each individual topic? Or should the backoff process take place in the overall model? If backoff takes place in the overall model, should smoothing be performed on each individual topic model? Or should we combine frequencies and smooth afterwards? This section will discuss the problems and benefits of each approach. We will use trigrams for our discussion, but the techniques are applicable to any order backoff ngram model.

3.1.3.1 Option 1: Individual smoothing and backoff The first option is to perform backoff and smoothing within each topic model. This is the most direct way to model the prior equations and is illustrated below:

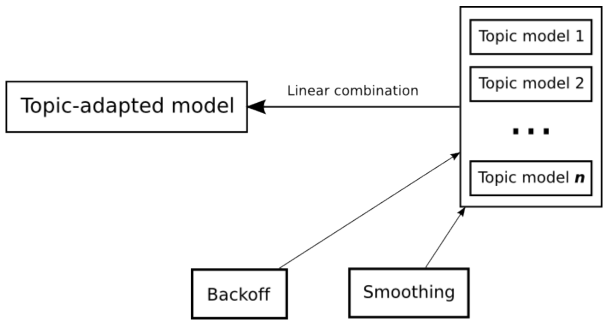


Figure 9: **Illustration of backoff and smoothing in the first implementation option.**

Additionally, we can represent this decision mathematically:

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

where $P_{\text{backoff}}(w \mid h, t)$ is Equation 4 with the additional conditioning information t:

$$P'(w \mid w_{-1}, w_{-2}, t) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}, t) & if P(w \mid w_{-1}, w_{-2}, t) > 0 \\ \alpha_{w_{-1},w_{-2},t} * P_{smooth}(w \mid w_{-1}, t) & if P(w \mid w_{-1}, t) > 0 \\ \alpha_{w_{-1},t} * \alpha_{w_{-1},w_{-2},t} * P_{smooth}(w \mid t) & otherwise \end{cases}$$

Smoothing is applied within this equation on each distribution for each topic.

The advantage of applying smoothing and backoff in this way is that the model fits nicely into the traditional approach of topic modeling (Equation 2). However, there are two significant problems with this approach. First, the held-out probability in smoothing will be computed on each (sparse) distribution. Probabilities of each word will be determined and interpolated using this overly conservative held-out probability mass. This will have the tendency to favor lower-order ngrams in testing when the interpolated distribution is in fact more reliable than the model accounts for. It is unclear how to remedy the problem with smoothing sparse distributions in this model. Second, the model doesn't offer the optimization that the following models offer. Generating the list of predictions can be thought of like a searching problem — attempting to search through the space of possible words to find the W most likely. However, in this model, very little constraint is offered on the word being predicted. Alternatively, if the backoff process were prestored for every word in the training vocabulary, the memory requirement would become impractical.

3.1.3.2 Option 2: Individual smoothing, combined backoff The second option is to perform smoothing on each individual topic model, interpolate the probabilities, and then perform backoff on the interpolated trigram, bigram, and unigram models, as in the following equations and picture.

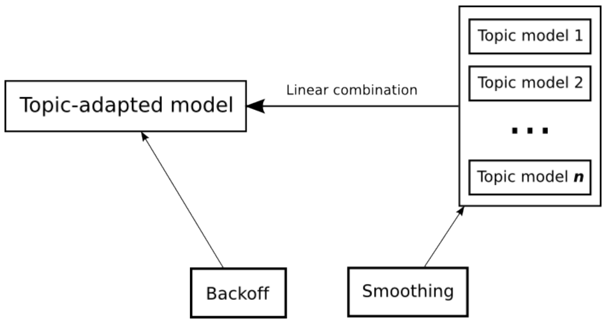


Figure 10: **Illustration of backoff and smoothing in the second implementation option.**

$$P_{\text{smooth}}(w \mid w_{-1}, w_{-2}) = \sum_{t \in topics} P(t \mid h) * P_{\text{smooth}}(w \mid w_{-1}, w_{-2}, t)$$

$$P_{\text{smooth}}(w \mid w_{-1}) = \sum_{t \in topics} P(t \mid h) * P_{\text{smooth}}(w \mid w_{-1}, t)$$

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

Smoothing is performed on each individual topic model, just like in Option 1, however, the smoothed probability distributions are combined and the resulting models are passed along to backoff:

$$P'(w \mid w_{-1}, w_{-2}) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1},w_{-2}} * P_{smooth}(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1},w_{-2}} * P_{smooth}(w) & otherwise \end{cases}$$

The backoff process computes each α_h either by subtracting the allocated probability over all words from 1 or by using a special symbol to store the held-out probability. However, as with the previous model, the held-out mass is overly conservative, because the amount of mass held out is computed on each of the topic distributions, which are more sparse than the sum of the topic distributions. Correcting the overconservative α's seems easier under this model than the previous one, but it is still a largely unexplored area of research to reduce the held-out mass in an interpolated model and then "unsmooth" this probability mass (simply scaling all words by this amount would be an incorrect inverse operation to smoothing in many cases). However, this model at least allows us to constrain the possible predictions to a smaller set than the whole vocabulary. The method to optimize this search is described in Appendix C.2.2.

3.1.3.3 Option 3: (Our approach) Combined smoothing and backoff The third option is to interpolate frequencies for each order model and then perform smoothing afterwards.

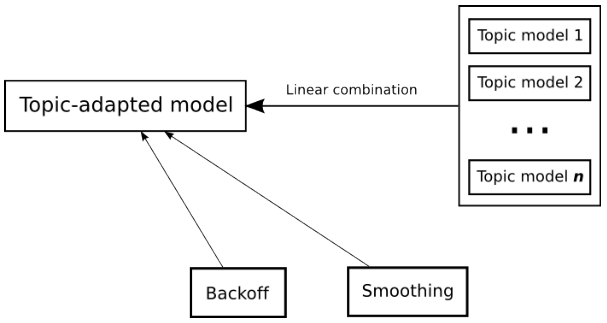


Figure 11: **Illustration of backoff and smoothing in the third implementation option.**

$$f(w \mid w_{-1}, w_{-2}) = \sum_{t \in topics} P(t \mid h) * f(w \mid w_{-1}, w_{-2}, t)$$

$$f(w \mid w_{-1}) = \sum_{t \in topics} P(t \mid h) * f(w \mid w_{-1}, t)$$

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

Once frequencies have been combined from each individual topic model, discounting is applied to create smooth probability distributions, which are fed into backoff below:

$$P'(w \mid w_{-1}, w_{-2}) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1},w_{-2}} * P_{smooth}(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1},w_{-2}} * P_{smooth}(w) & otherwise \end{cases}$$

The advantage of this approach is that the held-out probability for each distribution is appropriate for the training data, because the smoothing takes place knowing the number of words that occurred in the whole corpus, rather than for each little part. This is especially important when dealing with small topics, as in fine-grained topic modeling (Section 3.3). In addition, because large corpora are generally unavailable for AAC word prediction, we value the ability to address data sparseness very highly. Also, Option 3 allows the same optimization in generating the predictions as Option 2. Our ongoing research in topic modeling will focus on this third option, specifically to deal with data sparseness. Although researchers infrequently discuss these issues, Adda et al. [1] also chose to implement topic models by interpolating frequencies first, albeit for much larger topics than those in Switchboard and for a different task (broadcast news transcription). The following sections will address any issues in applying this method.

3.1.4 Smoothing real-values

Interpolating frequencies and then smoothing afterwards causes the frequencies to no longer be integers due to being multiplied with real-valued similarity scores. However, smoothing methods often rely upon integers — Good-Turing smoothing in particular considers the number of words that occur once, twice, etc. in order to smooth the distribution.

Our solution to this problem is to bin the real-valued frequencies and then number the bins, starting with one. Because the actual value of each bin is important in smoothing, we decided that the most reasonable solution was to create bins like so: $[0, 1)$, $[1, 2)$, $[2, 3)$ and so on. The implementation of this is a floor function with an addition.

3.1.5 Scaling frequencies

Another problem in the interpolation of frequencies is that the weighted average of the frequencies is much lower than the sum of the frequencies. For instance, after converting the real-valued frequencies to integers, most frequencies fall into the 0–1 range. If smoothing is applied on a distribution where most words are declared to occur once, it will reserve a large amount of the probability mass of the distribution of unseen words. However, in fact, many words in the distribution occurred multiple times and that should affect the way smoothing works.

To address this problem, we scale the distribution so that the sum of each conditional distribution is equal to the sum of the individual distributions, as shown below:

$$\sum_w f'_{topic}(w \mid h) = \alpha * \sum_w f_{topic}(w \mid h) = \sum_{t \in topics} \sum_w f(w \mid h, t)$$

where $f'_{topic}(w \mid h)$ is the new frequency used in the backoff process. In early studies using trigrams with topic modeling, we found that scaling the frequencies to sum to their original sum decreased keystroke savings by 0.1% at $W = 1$ and increased keystroke savings by 0.2–0.4% for window sizes 2–10.

3.1.6 Smoothing in backoff

Our initial experiments with smoothing were conducted on the baseline trigram model rather than the topic model for computational reasons. We originally used a variation of Witten-Bell smoothing [85] for simplicity, following these equations:

$$P(w \mid h) = (1 - P(unseen \mid h)) * \frac{f(w \mid h)}{f(h)}$$

where $P(unseen \mid h))$ is measured as the probability of a new word occurring in training. The MLE value is the number of word forms over the number of occurrences:

$$P(unseen \mid h) = \frac{\mid \{w \mid f(w \mid h) > 0\} \mid}{f(h)}$$

We later switched to the smoothing used in Katz' backoff[9] for high performance on the baseline trigram model [43]. The smoothing in Katz' backoff requires the entire trigram distribution to be smoothed at once.[10] However, a necessary optimization for topic modeling interpolates each conditional distribution only as probabilities are requested, allowing us to only smooth a small fraction of the entire distribution and reducing the computational

[9]Katz described a method for building language models using nonstandard smoothing and backoff, so when we say Katz' backoff we refer to a system using both techniques. Because it is a commonly accepted technique, we decided to evaluate the smoothing from Katz' backoff as well as the backoff part.

[10]This is because smoothing takes place on unconditional ngram distributions, not conditional distributions. When dealing with conditional distributions, they can be smoothed on-demand.

cost significantly. But this optimization prevents us from using true Katz' backoff. Instead, we set about to create approximations of Katz' backoff that could be applied to each conditional distribution individually. In the end, we settled on the following equation, as it is somewhat similar to Good-Turing smoothing [33], but without the requirement of a somewhat non-sparse distribution[11].

$$P(w \mid h) = \frac{f(w \mid h)}{f(w \mid h) + \lambda} * \frac{f(w \mid h)}{f(h)}$$

Originally, we used $\lambda = 1$ but later experimented with $\lambda = 0.5$ and found it to be a slight improvement. As with Good-Turing smoothing, each word is discounted, but the weight $\frac{f(w|h)}{f(w|h)+\lambda}$ discounts infrequent words by a larger percent than frequent words. This method is applied to backoff like Katz' method:

$$\alpha_h = 1 - \sum_w P(w \mid h)$$

$$P'(w \mid h) = \begin{cases} P(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1},w_{-2}} * P(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1},w_{-2}} * P(w) & otherwise \end{cases}$$

where α_h is the probability mass held out from the distribution $P(w \mid h)$ that is reserved for unseen words. A sample of the results of the three different methods is shown in Table 6 for early experiments on the Switchboard corpus without topic modeling.

Fringe word only				All words		
W	Witten-Bell approx.	Katz	Trnka approx.	W	Katz	Trnka approx.
1	282,652	282,307	282,576	1	830,796	829,910
3	220,512	219,857	220,080	3	659,013	658,490
5	201,706	200,946	201,174	5	595,700	595,990
7	190,940	190,319	190,497	7	557,458	558,031

Table 6: Number of keystrokes required under different smoothing methods. Lower is better. Prediction for fringe words only is shown to the left and prediction for all words to the right.

The initial comparison between the smoothing methods shows 1) that choice of smoothing method impacts the keystroke savings only a little (normally less than 0.1%) and 2) our approximation is better than Witten-Bell smoothing. While not quite as good as Katz' backoff in many cases, our approximation is better for small window sizes for all-word prediction and is compatible with our implementation of topic modeling.

3.2 Topic Identification/Relevance

Topic modeling can be viewed as a two stage process: 1) identifying the relevant topics and 2) tuning the language model based on relevant topics. In this section, we describe variations in identifying the relevant topics. This corresponds to $P(t \mid h)$ in the overall topic model (Equation 2), which we will refer to as a *(topical) relevance score*.

[11] Good-Turing smoothing computes discount ratios based on the ratio between the number of words that occurred r times and $r + 1$ times. However, in sparse distributions, we can't depend on this, as there tend to be a small number of word forms in each bucket. For example, we may have 3 words that occur once and 1 word that occurs 4 times. Good-Turing smoothing has trouble as there are no words that occur 2 or 5 times. The simple Good-Turing approach [29] deals with this by fitting a logarithmic curve to the data using linear regression and using the values of the fitted curve to "fill in" the actual data. This approach is useful for high values of r, relying on the low values of r to fit the curve. However, with very sparse distributions, the low values of r cannot be reliably used to make a curve.

Relevance scores are computed by comparing the unigram model of each topic with a unigram-like model of the conversation/document in progress. This model can be called a cache or recency model, and it is not restricted to pure frequencies. Generally speaking, the cache of the document is mapping of words to weights, where weights are determined using a combination of *frequency, recency, and topical salience* to model the topical importance of each word at the current point of the document. This cache representation of the document is then compared against the unigram model for each topic and then a relevance score for each topic is determined in order to compute the overall linear interpolation. This section first describes investigations in the cache representation of the document and then variations on the relevance score itself.

3.2.1 Document Cache

The cache of the current document is a model that maps individual words to weights, which are functions of each word's frequency, recency, and topical salience. In order to illustrate weightings for words, we present weightings for word *occurrences* and sum the weights for each occurrence of a term to compute the overall weight of the term.

The starting point for our investigation of the document cache ignores recency and salience, computing the weight of word w as the frequency the word in the part of the document encountered so far. The frequency of a word is computed by iterating over all words and adding one every time the desired term is encountered, illustrated for an example sentence in Figure 12. In this example, the weight for the term "Kathy" is 2 and the weight for other terms like "shared" is 1. This model, although simplistic, will weight the importance of words in proportion to

words	Kathy	shared	an	office	with	Kathy	in	g—
weights	1	1	1	1	1	1	1	

Figure 12: The user is currently typing "grad school" in this sentence. The weights are shown for each word occurrence.

how often the speaker or author uses them. However, the document cache can be improved by taking recency and topical salience into account.

3.2.1.1 Recency The recency of use of a word contributes to the relevance of the word. If a word was used somewhat recently, we would expect to see the word again. However, if a word was only used many sentences ago, we would not expect to see the word again compared to more recent words. These intuitions were confirmed by Beeferman et al. [4], who found that the probability of lexical repetition could be modeled using an exponential function of the recency of the word. We follow Bellegarda [7] in using an exponentially decayed cache to model this effect of recency on importance at the current position in the document. Each word occurrence is weighted differently in this method, illustrated in Figure 13. In this example, the term "Kathy" has a weight of 1.685 and "shared" has a weight of 0.774. In an exponential decay model, λ is a tunable decay weight. We chose $\lambda = 0.95$

words	Kathy	shared	an	office	with	Kathy	in	g—
weights	λ^6	λ^5	λ^4	λ^3	λ^2	λ^1	λ^0	
at $\lambda = 0.95$	0.735	0.774	0.815	0.857	0.903	0.95	1	

Figure 13: The user is currently typing "grad school" in this sentence. The weights are shown for each word occurrence, using recency. Frequency is implicit, as the weights of occurrences for each term are summed to weight terms.

following Bellegarda [7]. The effect of this model is that words that occur both frequently and recently are weighted highest. Words that occur infrequently are weighted lowly unless they occurred very recently, and words that occurred a few times, but not recently, are also weighted lowly. The weight of 0.95 models a preservation in topic, but with a decay for very stale words. It should be noted that a weight of 1 turns the exponential model into a pure frequency model, whereas a weight of 0 turns the model into a cache containing only the most recent word. A discussion of the benefits of exponential decay over linear decay or even a cache of the most recent k words can be found in [7].

3.2.1.2 Topical Salience The importance of each word occurrence in the current document is a factor of not just its frequency and recency, but also its topical salience — how well the word discriminates between topics. We would like to focus the scoring on words that give better topical discrimination and downplay words that are irrelevant to the topic.

For this reason, we decided to use a technique like TF-IDF to boost the weight of words that occur in only a few topics and depress the weights of words that occur in most topics. The TF-IDF measure has been shown to be a useful weight of discrimination power in other applications. However, instead of using Inverse Document Frequency (IDF) to measure topical salience, we use Inverse Topic Frequency (ITF). The main advantage of ITF is that it specifically weights words that are good discriminators between topics, which will be tailored to the particular topic granularity that we use (see Section 3.3 for topic granularity).

As is common in Information Retrieval, we take the log of the ITF. The intuition is that words that occur in only a few topics are weighted high and words that occur in every topic are low, focusing the similarity on words that differentiate one topic from another. An example of the weights with both recency and ITF is shown in Figure 14. For the purposes of simplicity, we replaced the ITF scores with approximated IDF scores from Google. In this example, the weight of "Kathy" is 10.151 and the weight of "shared" is 3.649. The weight of function words is much lower — "with" is assigned a weight of 0.930.

words	Kathy	shared	an	office	with
weights	$\lambda^6 * ITF$(Kathy)	$\lambda^5 * ITF$(shared)	$\lambda^4 * ITF$(an)	$\lambda^3 * ITF$(office)	$\lambda^2 * ITF$(with)
example	4.428	3.649	1.074	2.391	0.930

words	Kathy	in	g—
weights	$\lambda^1 * ITF$(Kathy)	$\lambda^0 * ITF$(in)	
example	5.723	0.406	

Figure 14: The user is currently typing "grad school" in this sentence. The weights are shown for each word occurrence, using recency and ITF. Frequency is implicit, as the weights of occurrences for each term are summed to weight terms.

Although using IDF-like methods should weight function words such as "an" or "the" very low (nearly zero), we prefer to simply use a list of common words that are independent of topic. In cases of small corpora and small topics, it is possible for IDF-like methods to accidentally weight uncommon stopwords highly, adding noise to the topic identification. For this reason, we ignored stopwords in processing using a list available online. In addition to this static list, we used a rule for a dynamic stopword list — words that occurred in 85% or more of topics were also ignored in building the cache. Because stopwords do not contribute to the weights at all in this model, we decided to also ignore stopwords in determining the recency component for each word occurrence. These changes bring us to the final version of our cache representation of the current document, illustrated in Figure 15.

3.2.2 Relevance Scoring

Now that the cache representation of the document in progress has been developed, we need to compare this model against a unigram model of each topic to approximate $P(t \mid h)$. In this section, we will first describe three

words	Kathy	shared	an	office	with	Kathy	in	g—
weights	$\lambda^3 * ITF(\text{Kathy})$	$\lambda^2 * ITF(\text{shared})$	0	$\lambda^1 * ITF(\text{office})$	0	$\lambda^0 * ITF(\text{Kathy})$	0	
example	5.165	4.256	0	2.649	0	6.024	0	

Figure 15: The user is currently typing "grad school" in this sentence. The weights are shown for each word occurrence, using recency and ITF. Frequency is implicit, as the weights of occurrences for each term are summed to weight terms. Stopwords are ignored in processing.

different functions to model $P(t \mid h)$ and then describe manipulations of the relevance functions to improve topic modeling.

3.2.2.1 Relevance functions Our initial work in approximating $P(t \mid h)$ used distributional similarity measures along with normalization (see [50] for a comparison of several measures of distributional similarity). Given an arbitrary function of the similarity between the topic unigram model t and the cache of the current document c, we normalize it to become a probability distribution:

$$P(t \mid h) \approx \frac{sim(t, c)}{\sum_{t'} sim(t', c)}$$

We initially used the cosine measure for similarity scoring, shown below. The geometric intuition of the cosine measure is that the topic unigram model and the cache of the current document are graphed in a V-dimensional space, where V is the vocabulary and the value of each dimension is the frequency of the corresponding word. In this representation, cosine is a measure of the angle between the two vectors. This measure accounts for the potentially different lengths of the two distributions, more akin to comparing probabilities between the distributions than comparing frequencies. The normalization for length is especially important when comparing a long and short document. In the long document, many irrelevant words may occur once, as if by chance. In the short document, even words that occur once may be important topic words.

$$sim_{cosine}(t, c) = \frac{\sum_{w \in t \cap c} f_t(w) * f_c(w)}{\sqrt{\sum_{w \in t} f_t(w)^2} * \sqrt{\sum_{w \in c} f_c(w)^2}}$$

where t represents the words in the topic unigram model, c represent the words in the cache, $f_t(w)$ is the weight of w in topic t, and $f_c(w)$ is the weight of w in the cache. Like most similarity measures, the cosine measure focuses on both the overlap between the two distributions as well as the overlap in relation to the size of each distribution. One of the appealing features of the cosine measure is that similarity is biased more towards matching on frequent words rather than infrequent ones.

Other researchers have demonstrated that cosine is not the best relevance metric for their applications [70, 50], so we decided to evaluate two other topic similarity scores: Jacquard's coefficient, which performed better than most other similarity measures in selecting the correct verb for an object for Lee [50] and Naïve Bayes, which gave better results than cosine for Seymore and Rosenfeld [70] for topic modeling.

The Jacquard coefficient is shown below. This method, although simplistic, was found to be the best previously existing technique by Lee [50] for differentiating between a artificial and natural verb-object pairs.

$$sim_{Jacquard}(t, c) = \frac{\mid t \cap c \mid}{\mid t \cup c \mid}$$

where t and c are sets of the words that occur in the topic and cache. Like the other methods, this one considers both the overlap between the two distributions and the non-overlap, but it ignores any weights associated with the words, instead computing the percentage overlap in the word forms.

We also investigated the Naïve Bayes method for relevance scoring. Whereas the similarity scores are very rough measures of $P(t \mid h)$ in conjunction with normalization, Naïve Bayes computes a true probability. The intuition behind Naïve Bayes is to assign the current document to a topic based on which topic unigram model assigns the highest probability to the document. In this way, words which are better identifiers of topics contribute more to topic classification. Therefore, we did not use ITF weighting in conjunction with Naïve Bayes. In addition to considering how words occur in topics, Naïve Bayes also considers the probability of a topic independent of the words (some topics may be very common and others very rare). The form of Naïve Bayes used for topic modeling is shown below (the common form would have a document d rather than a history h).

$$P(t \mid h) = P(t) * \prod P(w \mid t)^{f_h(w)}$$

where t is the topic, h is the history (the document seen so far), and $f_h(w)$ is the weight of the word in the cache representation of the document so far, just as with cosine. $P(t)$ is computed simply as the probability of any word in the corpus occurring in topic t and $P(w \mid t)$ is the probability of word w occurring in topic t, measured as a unigram model over the texts in topic t.

We used logs and scaled it down in order to use standard hardware:

$$\log(P(t \mid h)) = \frac{\log(P(t)) + \sum f(w) * \log(P(w \mid t))}{\sum f(w)}$$

We evaluated all three similarity metrics using Switchboard topics as the training data and each of our corpora for testing. The results were run using our cross-validation setup (11-fold, sets aligned across corpora, where cross-validation can be thought of as selecting columns of data and the evaluation specification selects the rows of data for training and testing). The table below shows the results of topic modeling for Switchboard-trained topics across corpora. Naïve Bayes is shown with and without scaling (which scales the maximum likelihood to 1 and the minimum to 0 before normalizing the scores, see Section 3.2.2.2 for more information).

Testing corpus	Cosine	Jacquard	Naive Bayes	Naive Bayes (no scaling)
AAC Emails	**43.527%**	43.179%	42.099%	42.213%
Santa Barbara	**43.902%**	43.454%	42.142%	42.385%
Callhome	**49.517%**	49.170%	48.317%	48.512%
Charlotte	**50.070%**	49.730%	48.195%	48.385%
Micase	**46.990%**	46.630%	45.412%	45.577%
Switchboard*	**61.478%**	60.636%	57.635%	58.089%
Slate*	**39.779%**	39.510%	38.207%	38.421%

Table 7: Switchboard topic modeling with trigrams compared with different similarity scores. Corpora with stars were too large to use cross-validation in time. The highest keystroke savings per corpus is shown in bold.

In our evaluation, we found that cosine is consistently better than both Jacquard's coefficient and both variants of Naïve Bayes . The differences between cosine and the other methods are statistically significant at $p < .001$.

The poor performance of Naïve Bayes was unexpected, especially since Naïve Bayes is commonly accepted as a baseline method in text classification [41] and was found to reduce perplexity slightly over TF-IDF for topic modeling for Seymore and Rosenfeld [70]. Even without the negative interaction between score scaling (see following section), Naïve Bayes still performs worse than cosine. It may be possible that the recency weighting in the cache also had a negative interaction with Naïve Bayes ; traditionally Naïve Bayes is applied on raw frequencies.

Jacquard performs much closer to cosine than Naïve Bayes did, though Naïve Bayes still performs significantly better. That Jacquard performs somewhat comparably to cosine is astonishing considering that it completely ignores all of the knowledge of frequency, recency, and salience and focuses solely on vocabulary overlap for topic identification. One of the possible explanations may be the out-of-domain nature of all tests other than Switchboard. In these cases, the vocabulary overlap between the training topics from Switchboard and the other corpora may be very small. In the case of very little vocabulary overlap between each topic and the testing data, both measures will produce small scores and we would expect them to perform similarly.

3.2.2.2 Score scaling One of the variations explored in Florian and Yarowsky [27] was to apply transformations on the similarity score. They found that this improved results using cosine similarity. Specifically, they found that applying the following function reduced perplexity more. This function first scales similarity scores so that the maximum score is 1 and the minimum score is 0, then squares the results.

$$sim'(t,h) = \left(\frac{sim(t,h) - min_{t'}(sim(t',h))}{max_{t'}(sim(t',h)) - min_{t'}(sim(t',h))} \right)^2$$

The purpose of scaling the similarity scores is to accentuate the difference between the scores. If a similarity score does not distinguish between relevant and irrelevant topics enough (e.g., if relevant topic scores are 0.5 and irrelevant topic scores are 0.45) then the topic model will not adapt to relevant topics as much as it should. Therefore, scaling the similarity scores can help fine-tune similarity scoring for topic modeling.

In their experiments, [27] found that squaring the score only reduced the perplexity slightly beyond the scaling, so we only perform the scaling and not the squaring:

$$sim'(t,h) = \frac{sim(t,h) - min_{t'}(sim(t',h))}{max_{t'}(sim(t',h)) - min_{t'}(sim(t',h))}$$

In our early experiments, we found that scaling the similarity scores in this way was beneficial when using cosine scores for topic identification. However, in evaluating Naïve Bayes with and without scaling (Table 7), we found that Naïve Bayes performs better without scaling. One of the possible explanations is that Naïve Bayes may already weight relevant topics very highly, and weighting them any higher over-focuses the model on the most relevant few topics. Also note, the Naïve Bayes method must be normalized after scaling to convert it back into a probability distribution.

3.2.2.3 Score smoothing One of the motivations behind using a linear interpolation of all topics is that the resulting ngram model will have the same coverage of ngrams as a model that is not adapted by topic. The topic adaptation method should tune probabilities to boost on-topic content and depress off-topic content (i.e., the baseline and topic models should have non-zero probability for *exactly* the same ngrams). For this reason, the resulting topic model should only perform worse than the baseline ngram model when it incorrectly identifies the topic of conversation. However, the similarity score will be zero when no words overlap between the topic and history. This is more likely when stopwords are filtered as well as at the beginning of the document, when the cache is very sparse. Additionally, due to score scaling, the similarity score is *guaranteed* to be zero for the least similar document.

Therefore we decided to experiment with similarity score smoothing, which would record the minimum nonzero score and then add a fraction of that score to all scores, then only apply upscaling — the maximum is scaled to 1, but the minimum is not scaled to zero. The scaling equation is modified below to include smoothing:

$$sim'(t,h) = \frac{sim(t,h) + \gamma * min_{t'|sim(t',h)>0}(sim(t',h))}{max_{t'}(sim(t',h)) + \gamma * min_{t'|sim(t',h)>0}(sim(t',h))}$$

where γ is a tuneable smoothing weight. When set to 1, all documents will have the minimum nonzero score added to their score. This has the effect of topics with zero score being assigned half of the score of the previous minimum nonzero score (half because all documents have this smoothing factor added, not just documents with zero similarity).

In pilot experiments, we found that this did not affect topic modeling with traditional document clusters. We hypothesize that the problem of zero scores in topics of this size is relatively insignificant, so smoothing was not necessary. However, with more sparse topics, as when each document is used as a topic (called fine-grained topic modeling in Section 3.3), we hypothesize that similarity smoothing will have a noticeable effect on topic modeling.

We experimented on fine-grained topic modeling with similarity smoothing and $\gamma = 0.3$. The results confirm

Testing corpus	**No smoothing**	**Smoothing**	**Significance**
AAC Emails	49.255%	49.375% (+0.120%)	$p < 0.02\%$
Santa Barbara	42.521%	42.574% (+0.053%)	$p < 0.001\%$
Callhome	43.600%	43.715% (+0.115%)	$p < 0.001\%$
Charlotte	48.574%	48.600% (+0.026%)	not significant
Micase	49.532%	49.554% (+0.022%)	$p < 0.01\%$
Switchboard*	61.435%	61.423% (-0.012%)	not significant

Table 8: Comparison between similarity smoothing with $\gamma = 0.3$ and no smoothing using fine-grained topic modeling (document as topic) and trigrams with 5 predictions across corpora on fringe words only. Cosine was used as the similarity function. Dictionary backoff was used. Cross-validation was used for all corpora but Switchboard.

that similarity smoothing is generally beneficial for fine-grained topic modeling. However, the results are not the same for each corpus — Switchboard shows a decrease in performance, although the change is not significant due to high variance. The Charlotte corpus shows an improvement, but due to the variance, the result is not statistically significant. Smoothing provides a small but significant benefit on the other corpora. The reason that the benefit is relatively small is most likely due to the least similar documents having zero similarity, in which case, the benefit of those documents is small (i.e., our similarity score was fairly good). It is likely that the benefit is vocabulary-related, where having non-zero weights for all topics allows the resulting model to have a complete vocabulary, in contrast to a model with some zero weights, where highly infrequent words may be accidentally pruned from the vocabulary.

3.2.2.4 Stemming Another alternative to improving similarity scoring is to stem the words in the topic unigram model and the document cache. Whereas before we discussed transformations on the relevance score itself, here we are discussing a transformation on the input to the relevance score.

We implemented stemming for cosine scoring using Porter's algorithm. We experimented on fringe words only using Switchboard, like many of our preliminary studies. We found that when fine-grained topic models were used (that is, each document is a topic, see Section 3.3), keystroke savings increased by about 0.2% across window sizes 1–10. However, we found that when using medium-grained topic modeling (human-annotated document clusters), keystroke savings *decreased* by 0.1–0.2% across window sizes. The results on fine-grained topic modeling follow our expectations — the similarity score is being performed on very sparse distributions, so stemming helps to lessen the sparse data problem. However, for medium-grained topics, where the data is much more reliable, stemming doesn't offer any benefit. In fact, stemming decreases performance using medium-grained topic modeling. We feel that this is the case because the stemmer may have blurred distinctions between different topics. For example, past tense conjugations of a verb may be associated with one topic while present or future tense conjugations may be associated with another topic. Additionally, Porter's stemmer may have incorrectly stemmed multiple words to the same root even though they may be from different root forms. A more accurate stemmer would address the second problem.

3.3 What's in a topic?

Our approach to topic modeling requires that the training data be segmented by topic. Documents in some corpora are manually assigned to topic labels, such as "air pollution" or "fishing" in Switchboard [74], which can be used for language modeling [77, 52]. Other researchers use topics of a similar type, but rely on automatic document clustering rather than manual clustering of topics [27]. However, other researchers have used each individual document as a sort of topic [58, 70]. At the other end of the spectrum, corpora tend to have very general themes, such as the Callhome corpus, which contains phone conversations between friends and family. The focus of this section is the difference between different levels of *topic granularity*, where fine-grained topics are generally very small, very specific collections of text (we take each document as a topic in this approach). On the other hand, coarse-grained topics contain a large amount of text and have a very general topic (e.g., news). Medium-grained topics are the traditional approach, where automatic or manual clustering forms topics that are reasonably specific (e.g., clothes, computers, sports). We implemented topic modeling at all three granularities, where we used documents as topics for fine-grained, Switchboard's topics for medium-grained, and corpora for coarse-grained (e.g., Switchboard, SBCSAE, Slate, Callhome). We evaluated all three approaches at $W = 5$ for a trigram baseline and a trigram-based pure topic model.

Not only do we vary the topic granularity, but we also control for corpus variations using our domain-varied evaluation of Section 2.2 and [75]. The relationship of the training data to the testing data describes the domain of the test. In-domain tests report the results of training using the same corpus as the testing data (on held-out data), which is the common evaluation for research. Out-of-domain tests report the results of training on all corpora except the testing corpus, approximating the real-world performance of the approach. Mixed-domain tests report the results of training on all corpora (on held-out data), similar to a real-world evaluation of a model that integrates testing data into training after processing.

The domain-varied evaluation interacts somewhat with the topic granularity information — not all combinations of domain and granularity are possible. Coarse-grained modeling takes an entire corpus as a topic, making in-domain evaluation uninformative. In this case, only one topic exists, reducing the topic model to the trigram baseline. Also, for medium granularity, we do not perform automatic clustering, so this evaluation is limited to training on Switchboard data (the only one of our corpora that has been annotated for topic). The medium-grained tests are an in-domain evaluation when tested on Switchboard and similar to an out-of-domain evaluation when tested on other corpora (as Switchboard is one of our largest corpora).

3.3.1 Medium-grained topics (human-annotated)

We evaluated medium-grained topic models using the manually annotated topics in Switchboard for training and each corpus for testing. The baseline method in this experiment is a non-adaptive trigram model trained on Switchboard. The results are shown in Table 9.

Testing corpus	Trigram	Medium-grained topic	Significance
AAC Emails	43.25%	43.53% (+0.27%)	$p < 0.05$
Santa Barbara	43.49%	43.90% (+0.41%)	$p < 0.001$
Callhome	49.33%	49.52% (+0.19%)	$p < 0.005$
Charlotte	49.64%	50.07% (+0.43%)	$p < 0.001$
Micase	46.52%	46.99% (+0.47%)	$p < 0.001$
Switchboard*	60.35%	61.48% (+1.12%)	$p < 0.001$
Slate*	39.17%	39.78% (+0.61%)	$p < 0.001$

Table 9: **Medium-grained, Switchboard-trained** Medium-grained topic modeling using human-annotated topics, trained on Switchboard. The baseline method is a non-adaptive trigram model trained on Switchboard. *cross-validation not performed due to time constraints

The test on Switchboard shows that medium-grained topic modeling is beneficial for in-domain testing. The remaining tests show that medium-grained topic modeling is even beneficial for out-of-domain evaluation. This demonstrates that topic modeling is significantly beneficial even when the training and testing are substantially different. The improvement due to topic modeling when testing out-of-domain is most likely due to two factors: 1) some of the topics of Switchboard may have occurred in other corpora and 2) topic modeling may have adapted using slightly relevant topics in Switchboard and "weeded out" obviously dissimilar topics. The second effect is the product of our topic modeling implementation — no lone topic is selected for language modeling, but all are allowed to contribute in proportion to their similarity. This method allows weak similarity to effectively fine-tune the language model to the current topic so long as the current conversation is remotely similar to some training data. This adaptability allows topic modeling to potentially fine-tune its predictions to a previously unseen topic of conversation, provided that the topic of conversation is remotely similar or dissimilar to some of the topics in Switchboard.

3.3.2 Coarse-grained topics (corpus as topic)

Coarse-grained topic modeling can be evaluated using both mixed-domain and out-of-domain training. In mixed-domain training, the training section of the testing corpus is included in the training data. In out-of-domain training, the training data from all other corpora is used. The baseline for comparison of both evaluations is a non-adaptive trigram model trained on the same data. The results of the mixed-domain evaluation are shown in Table 10 and out-of-domain shown in Table 11.

Testing corpus	Trigram	Coarse-grained topic	Significance
AAC Emails	52.18%	53.30% (+1.11%)	$p < 0.001$
Santa Barbara	47.78%	47.83% (+0.05%)	not significant
Callhome	53.14%	52.84% (-0.30%)	$p < 0.05$
Charlotte	53.50%	53.33% (-0.17%)	not significant
Micase	51.46%	51.56% (+0.10%)	not significant
Switchboard*	59.80%	60.62% (+0.82%)	$p < 0.001$

Table 10: **Coarse-grained, mixed-domain** Coarse-grained topic modeling using whole corpora as topics, trained on all corpora. The baseline method is a non-adaptive trigram model trained on all corpora. *cross-validation not performed due to time constraints

Testing corpus	Trigram	Coarse-grained topic	Significance
AAC Emails	47.89%	47.25% (-0.64%)	$p < 0.001$
Santa Barbara	46.97%	46.46% (-0.51%)	$p < 0.001$
Callhome	52.95%	52.52% (-0.44%)	$p < 0.005$
Charlotte	52.44%	51.82% (-0.62%)	$p < 0.001$
Micase	49.62%	49.23% (-0.39%)	$p < 0.001$
Switchboard*	53.88%	52.63% (-1.25%)	$p < 0.001$
Slate*	40.73%	40.48% (-0.24%)	$p < 0.001$

Table 11: **Coarse-grained, out-of-domain** Coarse-grained topic modeling using whole corpora as topics, trained on all corpora but the testing corpus. The baseline method is a non-adaptive trigram model trained on all corpora but the testing corpus. *cross-validation not performed due to time constraints

Coarse-grained topic modeling in a mixed-domain evaluation (Table 10) shows improvement in some cases and a loss of keystroke savings in other cases. However, for out-of-domain evaluation (Table 11), coarse-grained topic modeling is purely detrimental. The disadvantage of coarse-grained modeling may be due to two factors: 1) the corpora do not have a consistent but general topic or 2) the topic similarity score may correctly identify the topically appropriate corpora, but the stylistic differences between the corpora may dominate the results. The failure of coarse-grained modeling is difficult to interpret, but the results suggest that the coarse-grained topics were too general to be useful.

3.3.3 Fine-grained topics (document as topic)

Fine-grained modeling is the most flexible of the approaches, allowing us to perform all three domain-varied evaluations. We will present the results for in-domain evaluation in Table 12, out-of-domain evaluation in Table 13, and mixed-domain evaluation in Table 14.

Testing corpus	Trigram	Fine-grained topic	Significance
AAC Emails	48.92%	49.38% (+0.45%)	$p < 0.001$
Santa Barbara	42.30%	42.57% (+0.28%)	$p < 0.001$
Callhome	43.76%	43.72% (-0.05%)	not significant
Charlotte	48.30%	48.60% (+0.30%)	$p < 0.001$
Micase	49.00%	49.55% (+0.56%)	$p < 0.001$
Switchboard*	60.35%	61.42% (+1.07%)	$p < 0.001$

Table 12: **Fine-grained, in-domain** Fine-grained topic modeling using documents as topics, trained on the training sections of the testing corpus. The baseline method is a non-adaptive trigram model trained on the same corpus. *cross-validation not performed due to time constraints

Testing corpus	Trigram	Fine-grained topic	Significance
AAC Emails	47.89%	47.78% (-0.11%)	$p < 0.02$
Santa Barbara	46.97%	47.90% (+0.93%)	$p < 0.001$
Callhome	52.95%	53.18% (+0.23%)	not significant
Charlotte	52.44%	52.59% (+0.16%)	not significant
Micase	49.62%	50.69% (+1.07%)	$p < 0.001$
Switchboard*	53.88%	54.90% (+1.02%)	$p < 0.001$
Slate*	40.73%	41.19% (+0.46%)	$p < 0.001$

Table 13: **Fine-grained, out-of-domain** Fine-grained topic modeling using documents as topics, trained on all corpora but the one used for testing. The baseline method is a non-adaptive trigram model trained on the same corpora. *cross-validation not performed due to time constraints

In-domain evaluation (Table 12) shows that fine-grained topic modeling significantly improves keystroke savings for all corpora except Callhome. Out-of-domain evaluation (Table 13) presents a less clear trend, with fine-grained modeling significantly increasing keystroke savings for most corpora, but not for AAC Emails (significant decrease) or Callhome and Charlotte (insignificant increase). Mixed-domain evaluation (Table 14) shows a significant improvement for all corpora.

The results of the in-domain test on Switchboard are an interesting comparison to the medium-grained topics on Switchboard. The fine-grained approach offers 61.42% keystroke savings compared to 61.48% keystroke savings under medium-grained topic modeling (which uses human-annotated topics). The other medium-grained tests are not directly comparable to the out-of-domain evaluation for fine-grained modeling; even the baselines show substantial differences. We feel that the relatively minor difference in performance is encouraging for the fine-grained approach, which has the advantage of operating on unannotated corpora. The relatively mixed

Testing corpus	Trigram	Fine-grained topic	Significance
AAC Emails	52.18%	53.14% (+0.96%)	$p < 0.001$
Santa Barbara	47.78%	48.84% (+1.06%)	$p < 0.001$
Callhome	53.14%	53.39% (+0.26%)	$p < 0.05$
Charlotte	53.50%	53.92% (+0.42%)	$p < 0.001$
Micase	51.46%	53.13% (+1.67%)	$p < 0.001$
Switchboard*	59.80%	61.17% (+1.37%)	$p < 0.001$
Slate*	53.05%	53.66% (+0.60%)	$p < 0.001$

Table 14: **Fine-grained, mixed-domain** Fine-grained topic modeling using documents as topics, trained on all corpora. The baseline method is a non-adaptive trigram model trained on all corpora. *cross-validation not performed due to time constraints

improvements under out-of-domain training again seem to suggest that the style of discourse may have a more significant effect than the topic when the training data is both very different stylistically and very varied. Finally, we would like to discuss the keystroke savings offered under mixed-domain evaluation compared to in-domain evaluation. Theoretically, if the similarity scoring used in topic modeling is very good, we would expect the mixed-domain results using fine-grained modeling to always exceed the results under in-domain testing. We see that this is the case for the smaller corpora, where the baselines are substantially better under mixed-domain training (due to the much larger number of words in training). However, under Switchboard, the baseline is worse with mixed-domain training, as the amount of in-domain data is large compared to the amount of out-of-domain data. Unfortunately, we found that fine-grained topic modeling performs better under in-domain testing for Switchboard (61.42%) compared to mixed-domain testing for Switchboard (61.17%), though the mixed-domain results are still higher than the in-domain baseline. This suggests that our similarity metric could potentially be improved, or that style modeling may account for any deficiencies in our approach with mixed-domain training.

3.3.4 Nearest-neighbors approaches

Fine-grained topic modeling can be used in conjunction with a nearest-neighbors approach. Mahajan et al. [58] selected the most similar 50, 100, 200, and 400 documents for four topic-adapted language models, however, individual documents were not weighted, but the groups of the most similar 50, 100, 200, and 400 were weighted. In our work, this approach makes the most sense for fine-grained topic modeling. Fine-grained topic modeling is computationally demanding, so our original implementation focused on the k-nearest neighbors (knn) approach, where the most similar k documents are selected (similar to Mahajan et al.) and then they are weighted based on their relative similarities (in contrast to Mahajan et al.). We performed experimentation with various values of k to maximize keystroke savings and found that $k = 1250$ was a good setting (out of 2,217 documents in Switchboard). The evaluation of different values of k is shown in Figure 16.

However, the trend changed when we applied stemming in similarity scoring. Stemming improved keystroke savings for fine-grained topic modeling across window sizes (see Section 3.2 for more details). Not only did stemming improve fine-grained topic modeling, but it changed the nearest neighbors trend. Figure 17 shows the new trend — keystroke savings is maximized when all 2,217 documents are used in fine-grained topic modeling with stemming. Note that we only evaluated values of k for which maximum keystroke savings was achieved at some window size, as the prior evaluation demonstrated that small values yielded poor keystroke savings. We feel that this demonstrates an interesting evaluation for similarity metrics — with a good similarity measure, keystroke savings should never decrease as the number of nearest neighbors increases. The trend should see diminishing returns as more neighbors are added; after all, the last documents to be added should be the least helpful (and conversely, the first documents to be added should be the most helpful). On one hand, these results demonstrate one characteristic of a good similarity metric. On the other hand, they demonstrate that when the quality of the

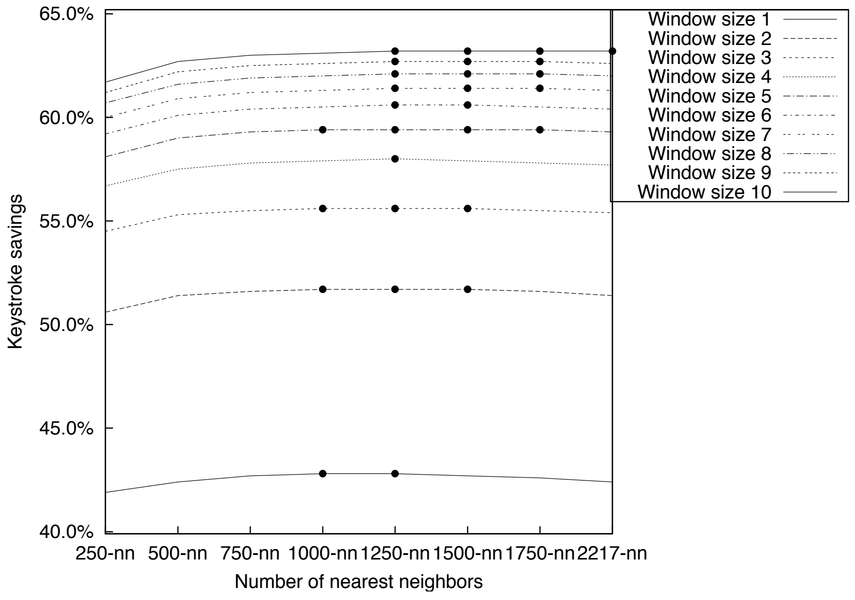


Figure 16: Keystroke savings across window sizes at different numbers of nearest neighbors. The maximum value for each window size is shown with a dot. When multiple window sizes have the same value to one decimal place, there are multiple dots per line.

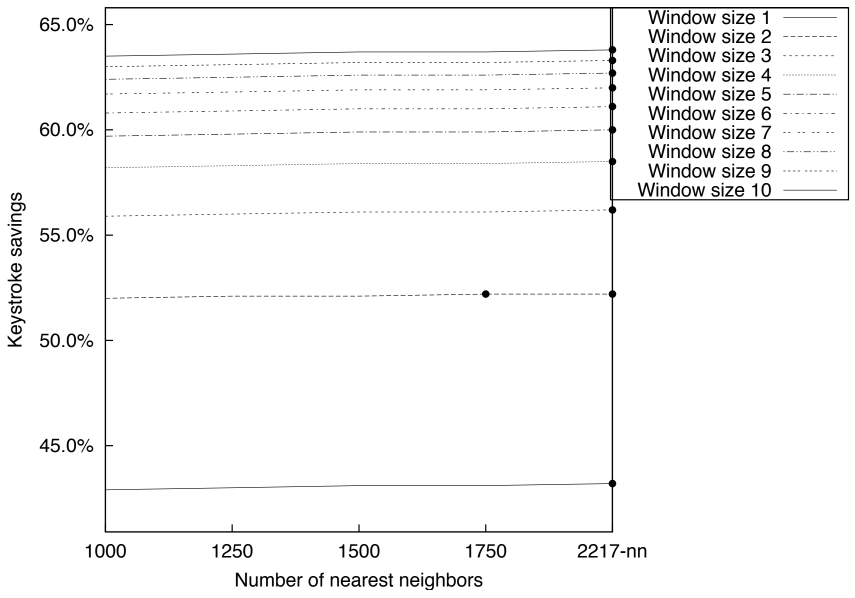


Figure 17: Keystroke savings at across window sizes at different numbers of nearest neighbors using stemming. The maximum value for each window size is shown with a dot. When multiple window sizes have the same value to one decimal place, there are multiple dots per line.

similarity metric is unknown or questionable, it may be advantageous to tune k using held-out data.

3.3.5 Related work in topic adaptation

Adaptive methods rely on the part of the document that has already been processed (or document history) and seek to fine-tune the language model to the document history. However, approaches in topic adaptation differ in the method of fine-tuning the language model to the document history. Methods generally fall into three categories of topic adaptation.

topic models operate on corpora that are segmented by topic. In testing, they seek to tune the language model to the most relevant topics

Latent Semantic Analysis (LSA) models train a model that associates words with abstract semantic concepts, and then in testing seeks to tune a baseline language model to be more similar to the semantic concepts of the document history

trigger pair models identify how certain words (triggers) increase the likelihood of other words (targets) and then in testing boost probabilities of target words for triggers in the document history

In the following sections, we will describe related work in topic adaptation for word prediction and speech recognition, covering each of the three categories of topic adaptation separately. We will focus on the most relevant work first and particularly focus on the adaptation of the language model itself.

3.3.5.1 Topic modeling Approaches in topic modeling operate on training texts that are explicitly segmented into topics and focus the language model on the most relevant topics, usually by leveraging a topic relevance score [70, 69, 16, 58, 27, 18, 31, 38, 1, 45]. Most researchers train a separate ngram model on each topic in training, but differ in the way that topic adaptation is performed and how to decide which topics are most relevant to the current document. The three main methods for computing the topic relevance score are the Expectation-Maximization (EM) algorithm, Naïve Bayes classification, and cosine similarity of unigram distributions (potentially weighted in a TF-IDF manner). The EM family of scores uses an optimization algorithm to iteratively calculate ideal scores, applied to topic modeling by [18, 31, 38, 1]. The Naïve Bayes method fits more closely with traditional research in topic classification, used by [70]. The cosine score approach takes an Information Retrieval mindset and has been applied to topic modeling by [70, 58, 27]. Although we are not aware of any research comparing the three scores directly, all three have been applied in successful topic adaptations.

Lesher and Rinkus [52] perform pilot research for topic modeling with word prediction on the Switchboard corpus, which is segmented by topic. In training, they build an ngram model for each topic individually. In testing, they select the ngram model for each testing conversation's topic, researching the sort of performance improvements possible if perfect topic identification is performed. The results under this approach were disappointing however, as a single model constructed from all training data gave better keystroke savings than selecting the appropriate topic model, due to the much larger size of the overall model. A blending of the overall model with the topic-specific model showed improvement — 58.74% keystroke savings for the "Big+Auto" model compared to 57.81% for the "Big" model alone. Their work demonstrates that topic modeling has the potential for improving word prediction, however, their approach is only preliminary, as it assumes that the correct topic is provided to the system. In addition to the problem of manually identifying the correct topic, their approach also assumes that the best topic-adapted language model treats all topics but the correct one equally. However, many topics other than the most relevant topic can be useful, especially if there are several similar topics.

Seymore and Rosenfeld [70] built topic-adapted language models for speech recognition from a segmented corpus using linear interpolations of individual topic models. In their initial experimentation, they compare the current document to each topic and compute a relevance score for each topic. The highest scoring 5, 10, or 20 topics are selected for adaptation. An ngram model is computed for each group of relevant topics (i.e., 5, 10, 20) and then the overall model weights each of these components in addition to a general-purpose ngram model. The interpolation weights in the overall model are determined using the EM algorithm. One of the variations in their research is the choice of relevance score function for comparing the document to each topic. The first method ranks topics by the cosine of TF-IDF values between each topic and the current document, similar to our (unnormalized) topic identification score. The second method uses a traditional Naïve Bayes classifier using each topic as a text class. They experimented with all combinations of relevant topics (5, 10, or 20) and relevance score (Naïve Bayes , TF-IDF/cosine), finding that all six variations decreased perplexity over the general-purpose baseline model. Naïve Bayes decreased perplexity more than TF-IDF with cosine scoring and they found that perplexity was lower when 20 topics were selected as opposed to 10 or 5 (also, 10 was better than 5). In contrast, when we experimented with Naïve Bayes compared to cosine, we found that the cosine score gave higher keystroke savings (see Section 3.2 for more details). In additional experiments, they tried to address the potential sparseness of each individual topic by merging small topics with their parents in an automatic hierarchy of topics (230 of their 5,883 original topics contained fewer than one thousand words). They applied techniques to generalize a topic to its parent when the number of words in the topic were below a threshold. In the evaluation of tree-based generalization techniques, again they found that Naïve Bayes was better. With their best techniques, however, perplexity was the same as the original non-hierarchical approach. Their work in hierarchical modeling addresses similar problems to our usage of weighting each topic to create a model that is both adapted, but also with enough data to allow a reasonable language model to be constructed. Seymore and Rosenfeld also evaluated using word error rate in an N-best rescoring system, though the interpolation weights were slightly different. Because the actual conversation isn't known (only the error-laden speech recognizer output), they experimented with using the EM algorithm to minimize the perplexity of the speech recognizer output as well as assigning the general model a weight of 0.55 and giving equal weights to all other models. The weights that minimized perplexity resulted in an equal or lower WER for both the development and testing sets across different adaptation methods. The best method on the development data was the original Naïve Bayes approach without any tree-based generalization (39.5% WER compared to 39.6% for TF-IDF and Naïve Bayes with orphan trees, all with min-PP weights). The best methods on the testing data were the opposite — TF-IDF without tree-based generalization and Naïve Bayes with the orphan trees had a WER of 35.3% (compared to 35.4% for the original Naïve Bayes approach). The best approaches yielded a 0.7% reduction in word error rate for development data and 0.1% reduction for testing data. Although they apply use TF-IDF with cosine scoring to identify relevant topics, their research differs significantly from ours. In their approach, a fixed number of topics are selected for topic adaptation. However, because the size of topics may vary, this approach can create a topic model that is computed from a very sparse on-topic model and a robust baseline model. In this case, the on-topic model offers little benefit over the baseline model due to data sparseness. In addition to the problem of selecting a fixed number of most relevant topics, they weight all relevant topics equally. However, it may be the case that only some of the selected topics are relevant, in which case, the model will be washed out by somewhat irrelevant data. We feel that our approach to pure topic modeling integrates both topic-adapted information as well as baseline information in a manner that is more flexible to situations in which very few topics are relevant or many topics are relevant.

In further work, Seymore et al. [69] experimented with a non-linear interaction between general words, on-topic words, and off-topic words. The system built a topic model using the five most similar topics, where the most similar topics were determined using a Naïve Bayes classifier. They first segment the vocabulary into general and topical words using Hotelling's T^2 test and Kullback-Leibler distance. After the 5 most similar topics are selected, they segment the topical words into on-topic and off-topic words using the χ^2 test as well as average mutual information. Then when the probability of a word is requested by the speech recognizer, if the word is in the general vocabulary, the probability from the general model is used. If the word is in the on-topic vocabulary, the probability from the 5 topics is used, multiplied by a weight. If the word is in the off-topic vocabulary, the probability from the general model is used multiplied by a weight. The on-topic and off-topic weights are formulated so that the sum of probabilities of on-topic and off-topic words are identical in the general model and the new approach (this ensures that the combination model sums to 1), and also weights are formulated to down-weight off-topic words and boost the probability of on-topic words. They evaluated various combinations of how to select general words and on-topic words, but found that none of the resulting models lowered perplexity more than a linear interpolation (though the non-linear models still reduced perplexity over the general baseline language model). While the explicit segmentation of the vocabulary into on-topic, off-topic, and general words is an interesting approach to the problem of topic adaptation, it is unclear how to improve the approach to have a more beneficial effect.

Chen et al. [16] extended the idea of boosting the probabilities of on-topic words and depressing off-topic words by integrating the topic adaptations into an exponential language model, similar to the Maximum Entropy model of Rosenfeld [65] (to be described in the trigger pairs section). Because the normalization step of the exponential model is computationally demanding, they omitted normalization. Their scores are not probability estimates, making perplexity values unavailable, but the scores can be used in a speech recognizer. They found that this kind of topic adaptation in an exponential language model reduced word error rate by 0.5% (from 30.8% WER to 30.3% WER). One of their additional findings was that depressing off-topic words contributed little to the improvements, so their final experiments only boosted on-topic and article-specific probabilities. It is unclear how to compare their results to ours.

Mahajan et al. [58] sought to topically adapt a language model by applying information retrieval techniques. Their philosophical motivation aligns with ours — that real-world text will not necessarily fit nicely into any one predefined topic. Therefore they sought to not only allow multiple topics to be "active", but also worked at a fine granularity to allow topics to be as specific or general as the application demands. Their main approach uses the TF-IDF measure to rank *documents* in the training data based on their cosine similarity to the first 100 words of each testing document. The language model for the top few documents is likely to be highly related to the testing document, however, the data may be somewhat sparse, so a larger topic-centered language model may be better. This motivated experimentation with the four different thresholds for topically related documents. The top 50, 100, 200, and 400 ranked documents are each used to create separate topic-centered language models. They note that it may be possible to weight each document according to it's cosine similarity, which originally motivated us to pursue our fine-grained topic modeling. They experiment with a large linear combination framework, where many different language models are combined with optimized weights derived using the EM algorithm on the first 100 words. The remainder of the words in the document are used to evaluate the combination model, measured in perplexity. New words were excluded from perplexity calculations.[12] The baseline trigram model is included in every variation of the approach and results are reported in perplexity as well as reduction over the baseline model. They tried adding a cache model to the baseline model and found that it reduced perplexity by 14.9%. The cache was populated using the most recent 500 words of the document and was not weighted by distance. Mahajan et al. first tried to incorporate the dynamic topic model (built from the highest ranked documents) by using the 100 most similar documents and adding this model into the linear interpolation. They found that this reduced perplexity by 32.3% over the baseline model. They also tried having four dynamic models (50, 100, 200, and 400 documents) as four separate components in the linear interpolation (along with the baseline and cache models). This approach reduced perplexity further, giving a total perplexity reduction of 35.2%. For comparison, they also built 10 static topic clusters of the training data using existing clustering methods and also 20 static clusters. They found that these approaches reduced perplexity over the baseline method by 28.0% and 28.1% respectively. However, the dynamic models reduce perplexity more. To improve the quality of the approach, they try applying stemming to the words before computing the TF-IDF scores for ranking. They found that this improved the dynamic model further, giving a total perplexity reduction of 36.0% compared to 35.2% for the same method without stemming. This is similar to our findings with fine-grained topic modeling (Section 3.3) — individual document are often too small to have reliable unigram distributions for topic identification, so stemming the distribution addresses the data sparseness problem. They finally tried combining the four dynamic models with the 10 static topic models with stemming in the framework (the baseline and cache models are still included) and found that the total perplexity reduction was 37.6% compared to 36.0% without the 10 static models. They also investigated other Information Retrieval techniques, such as stopword removal and query expansion, however

[12]Although this is uncommon in the speech recognition community, it is a fair evaluation for improvements in language modeling as the results are formulated in terms of percentage improvement rather than comparing the absolute perplexity values against other researchers. However, the perplexity reductions may seem overly large compared to other researchers in part because new words were excluded.

they found that they did not improve the document ranking. The first major difference from our work is that we use dynamic topic modeling, which iteratively refines the topic model as the testing data is encountered. This avoids Mahajan et al's assumption that the first 100 words of the document are available for tuning. Especially in the case of smaller documents such as emails, we feel that this is an unrealistic assumption for word prediction. Also, their approach in selecting the most relevant text echoes Seymore and Rosenfeld [70]. However, weighting each document by relevance allows for much more flexibility and avoids the need to fine-tune the size of the list of most relevant topics for each task. Although they achieve a larger reduction in perplexity than Seymore and Rosenfeld, it is unclear how this would translate to our work in word prediction.

Florian and Yarowsky [27] explore the usage of hierarchical topic clustering for to adapt language models for speech recognition. They use a traditional agglomerative clustering technique on broadcast news text to build their initial topic hierarchy and then fine-tune the approach for language modeling. In training the hierarchical language model, frequencies are collected for each document in each cluster first and then they are propagated up in the tree, so that the root is identical to a smoothed model measured from the entire corpus. Then a language model is setup for each node in the tree, where the language model is a linear interpolation of three components: 1) the probability of the word in the node in the tree, 2) (recursively) the probability of the word in the parent node in the tree, and 3) (recursively) the probability of the word in the node of the (n-1)gram version of the tree. The interpolation weights for the three components was static when they experimented with a bigram model, and dynamically optimized using the EM algorithm when they experimented with a trigram model. The underlying intuition of the model is that using a more general topic and using a model with less conditioning information are two different ways of backing off to less sparse, less constrained language models. In testing, a similarity score is used to approximate $P(t \mid h)$. This value seems to be computed for each leaf in the tree. They used the cosine similarity between each topic and the current document history to compute the likelihood of each topic, and experimented with similarity score scaling as well as squaring the resulting score. The final language model seems to be a linear interpolation (similar to our equation), but using the interpolated language model of each leaf node rather than a simple ngram model. The results show that scaling the similarity scores and squaring the result were the best methods, both individually and together. This motivated our usage of applying functions to the similarity scores in Section 3.2.2.2. The best combination of these techniques in a bigram model reduced perplexity by 10% overall and 33% on the target word set (topical words). Their approach is similar to our approach, with the exception of integrating topic generalization into backoff language modeling. However, it is unclear that the generalization approach to integrating topics is necessary. It may be the case that their doubly recursive backoff process addresses the same issues as interpolating frequencies. We feel that our approach is better suited for extending to include style adaptations as well as topic adaptations.

Several researchers have applied the Expectation-Maximization (EM) algorithm to finding optimal values for the interpolation weights for topic modeling [18, 31, 38, 1]. The basis of this approach is the usage of a hill-climbing algorithm to find weights that maximize probabilities. In testing, the EM algorithm for topic modeling is used to periodically update the interpolation weights based on the document history. The approach iteratively computes the expectation of its variables and then adjusts parameters to maximize the likelihood of the conversation. In this way, it optimizes the overall language model's probabilities for the each conversation encountered in testing. However, due to the automatic learning nature of the EM algorithm, it is unclear whether it captures topic, style, or some combination of the two.

Clarkson and Robinson [18] build a trigram model that uses the EM algorithm to adapt to the current conversation for speech recognition. In training, the British National Corpus (BNC) is clustered into different topics (both manual and automatic methods were tried) and a separate trigram model is created for each topic. In testing, the EM algorithm is used to derive weights for a linear interpolation of all the individual topic models. After every 10% of the conversation, the weights are derived again to optimize the combination model for the current conversation. Also, they added a "topic" containing all training data into the model to balance the data sparseness introduced by segmenting the data many ways. They found best results with 50 topics using the general-purpose topic containing all training data; these methods reduced perplexity by 24.0%. They also investigated the addition of a cache component using exponential decay, similar to the model we used to compute similarity scores for topic modeling. They found that the cache component reduced perplexity by 14.4% over the trigram baseline and that the combination of both topic adaptation and the cache component gave a total reduction of 30.0% perplexity. Although they intend to adapt to the topic and style of conversation, it is unclear what exactly the EM algorithm is adapting to. Furthermore, it is unclear how to improve the adaptations in the EM framework, whereas our approach allows the researcher to improve topic identification by improving the similarity score.

Gildea and Hoffman [31] use the EM algorithm to adapt a unigram distribution to the topic of conversation for speech recognition, similar to our hybrid approach with a static trigram model and a topic-adapted unigram model. They experimented with different ways of combining the static trigram model and unigram topic model, finding that geometric combinations gave lower perplexity than linear combinations. The resulting model lowered perplexity on the testing data by 16% for a static trigram model compared to a trigram baseline and 20% for a static bigram model compared to a bigram baseline. The topic-adapted model was found to decrease word error rate (WER) on rare words as desired, but it increased WER on frequent words, leading to an overall increase in WER despite the decrease in perplexity. The breakdown between perplexity and word error rate is explained in Chen et al., 1998 [17]. Their approach is similar to our hybrid topic model. Their findings that the topic model did not reduce WER overall (despite the improvement on rare words) is similar to our finding that hybrid topic modeling was limited by the combination of the topic-adapted unigram model with the baseline context-based model. We feel that the problem of a hybrid topic model has been addressed by focusing on pure topic modeling.

Iyer and Ostendorf [38] apply topic mixtures to score whole sentences for speech recognition. In contrast to our research in topic modeling, they compute the probability of a whole sentence using each individual topic model and afterwards use a linear combination of the probabilities from each topic model to compute the final probability of the sentence. Their goal is to use a topic-based mixture to capture the topical coherence within a sentence and to apply a cache-based component in the framework to capture the topical coherence within an article. They initially cluster sentences from the training data into topics by using a score based on word overlap with agglomerative clustering, and later refine the topic clustering by using the EM algorithm to help re-group sentences by trigrams rather than just unigrams. In testing, the values for the linear interpolation weights are computed on held-out data once. They note that dynamic re-estimation of the weights is possible, but that they found dynamic adaptation of the mixture weights did not improve perplexity or word error rate. In contrast, our topic identification scores are recomputed often to reflect the changing knowledge about the topic of conversation. The cache component is integrated into the mixture model by assigning each sentence of the current document to topic models based on the likelihood it belongs to each model, then when computing the probability of each sentence, the static ngram probabilities are linearly interpolated with the cache-based probabilities. The two techniques moderately decreased word error rate over a trigram baseline (11.5% WER with trigrams to 10.8% WER with both methods). Their treatment of a topic as clusters of related sentences is different than our clusters of documents. Consider all sentences containing the word "ball" compared to all documents containing the word "ball". We would expect the sentence clusters to be much more similar to one another than the document clusters. However, this limits the topic generalization of the approach. Topic modeling is intended to include as many relevant sentences as possible, not just sentences that have keyword overlap. Additionally, their approach focuses on a whole-sentence language model, which is applicable to re-ranking candidate sentences for speech recognition, but not applicable to word prediction.

Adda et al. [1] address the problem of using three very different training corpora for broadcast news transcription. They use the EM algorithm to determine weights for a linear interpolation of models from each of the three corpora in a very similar manner to our topic modeling, except that the data used the EM algorithm was the eval97 dataset. In this sense, they are not adapting to the specific topic of each conversation, but the overall topic of the test genre, similar to our coarse-grained topic model (see Section 3.3). The approach was evaluated on eval96, eval97, and eval98 separately. They also experimented by varying the corpora used in the interpolation. Like other researchers, they found that the perplexity results did not match the word error rate (WER) results well, even to the extent that perplexity increased in some cases while word error rate decreased. They found that the best results were achieved when all three corpora were used in the interpolation rather than several individual corpora or pairs of corpora. This approach is similar to our coarse-grained modeling. While they improve news transcription, it is questionable whether such a coarse granularity in combination with their corpora (different years of data from the same source) is modeling topic.

Kneser and Steinbiss [45] apply the forward-backward algorithm (an instance of the EM family of algorithms) to a linear interpolation of individual language models. Their topics are akin to genres, such as newspapers, scientific texts, and fiction. They initially found that optimizing the weights for each category in testing reduced perplexity from a baseline of 532.1 to 487.6 (this gives a sort of lower bound on perplexity if the testing category is known a priori). They found that optimizing the weights for the previous 400 words after every word reduced perplexity to 480.7 (adapting to each specific document gave a lower perplexity than optimally adapting to each document's topic). They also experimented with a hard decision about the topic of the testing conversation by using the model of the most likely topic, but found that this approach was inferior with a perplexity of 494.2. The authors added a cache model into the system and found that it further reduced perplexity to 384.0 (and that both the cache and topic modeling components contributed to the overall reduction). As with the Adda et al. [1] this approach has very general topics. Also, like other research that applies the EM algorithm to topic modeling, it is unclear how to improve or tune the method. Otherwise, their approach is similar to ours, but the results are not comparable.

3.3.5.2 Latent Semantic Analysis Latent Semantic Analysis (LSA) is an application of matrix math to machine learning which attempts to uncover the hidden semantic relationships between words, originally designed for document retrieval by Deerwester et al. [21][13]. The input to LSA is a word by document matrix and Singular Value Decomposition (SVD) is applied to convert the simple word-document matrix into the product of three matrices — a word by semantic dimension matrix, a semantic dimension by semantic dimension matrix (the only non-zero values are along the diagonal, representing the salience of each dimension), and a semantic dimension by document matrix. The SVD procedure is designed to create this three matrix product to represent the original matrix. The semantic dimensions resulting from SVD can be intuitively thought of as concepts, although the mathematical procedure does not explicitly require them to represent concepts. The middle of the three matrices contains weights for each semantic dimension along the diagonal and zeros in all other positions. These weights characterize the importance of each semantic dimension. The LSA approach simplifies the overall decomposition by dropping the least important semantic dimensions from the middle matrix in accordance with user specifications, treating the least significant dimensions as "noise" in the data. Most approaches have the researcher or user specify the desired number of semantic dimensions manually and all but the most important k dimensions are dropped from the decomposition. Deerwester et al. [21] used 100 dimensions for document retrieval in a relatively small corpus (1000–2000 documents, 5000–7000 terms). Dumais et al. [25] applied LSA to cross-language document retrieval using 330 semantic dimensions. Landauer et al. [48] experimented with 2–1032 dimensions in applying LSA to the TOEFL vocabulary test and found that performance peaked with 300–325 dimensions. The optimal choice of k may vary from one application to another, but researchers generally select values in the hundreds.

Wandmacher and Antoine [82, 84] applied LSA to language modeling for word prediction. They train a baseline 4-gram model on 44 million words of French news text using the SRI toolkit[14] and train the LSA model on 100 million words of news text using the Infomap toolkit.[15] The LSA space was reduced to 150 semantic dimensions in training. In testing, the semantic vector for each word (the vector comes from the word by semantic dimension matrix from LSA) is added to a tally as words occur. This gives an LSA representation of the current document without having to re-do the entire semantic analysis to include the current document.[16] The probability of a word given a history for the LSA-based component is computed by taking the cosine between each word and the semantic representation of the current document. The cosine score is scaled to improve discriminative power and then normalized so that the scores meet the criteria for probability distributions (similar to our approach with cosine scores). Wandmacher and Antoine experiment with various ways of integrating semantic information with a standard ngram model and a unigram cache model. In general, they found that a geometric combination of the

[13]The application of Latent Semantic Analysis (LSA) to information retrieval is often called Latent Semantic Indexing (LSI), following this work.

[14]`http://www.speech.sri.com/projects/srilm/` as of 12/11/2007

[15]`http://infomap-nlp.sourceforge.net/` as of 12/11/2007

[16]This approach has also been used to convert a query into a pseudo-document for information retrieval.

LSA component with the 4-gram component was better than a linear interpolation. They also found that using dynamic interpolation weights based on confidence measures was better than static weights. The cache with the 4-gram alone did not improve keystroke savings nearly as much as the LSA techniques. Even when the cache was generalized by using LSA to bring in words related to cache words, the results were still worse than the LSA models with the 4-gram model. The conclusion was that Latent Semantic Analysis could be used to improve keystroke savings, especially when a confidence-weighted geometric interpolation is used to combine the ngram and LSA components (+1.05% keystroke savings). Their improvement in keystroke savings is roughly comparable to our findings using topic modeling. However, the LSA approach is very computationally demanding in training, so the LSA component in not practically applicable to retraining on user data. In contrast, because topic modeling performs generalization at testing time, user data is more easily integrated into a topic model compared to an LSA model. Another difference from our work is that they focus on written text, whereas our focus is on spoken text. This difference is reflected not only in the genres used for training/testing, but because they use written text, they are able to utilize much larger corpora for training, which in turn affects the methods they can apply.

Bellegarda [7, 5, 6] applied LSA to language modeling for speech recognition by creating an LSA based probability distribution and combining it with a standard ngram model. The combination model multiplies the probabilities of the baseline ngram model with the LSA probabilities. The ngram model contributes word preferences based on the immediate context (the previous few words), while the LSA model contributes preferences based on the entire document (words that bear semantic relations to the document). This combination can be viewed as a geometric combination, but without weights. This combination model inspired our experiments in hybrid topic modeling, but we found that weights were necessary (see Section 3.1.1). The LSA component was experimented with in detail, particularly whether word/document clustering was used. The baseline LSA model converts the document history into the LSA space by summing the LSA vectors for each word in the document history and normalizing (similar to treatment of the query in LSI). Words are then assigned probabilities as the normalized cosine similarity between their semantic representation (the LSA vector for each word) and the LSA version of the document history. The normalization divides each cosine similarity by the sum of all similarities to convert the similarity scores into a probability distribution. Bellegarda [6] gives further detail regarding the LSA approach, showing that the original word-document matrix passed to LSA contains more than simple frequencies, but complex weights similar in spirit to TF-IDF weights. The direct adaptation approach was trained and tested on different parts of the NAB News corpus (42 million words in training, 2 million testing). They found that it reduced perplexity from a bigram baseline of 215 to 147 with a bigram model combined with the LSA model (a 32% reduction) [5]. A standard trigram model gave a perplexity of 142. Further work showed that a trigram model combined with LSA gave a perplexity of 115 (19% PP reduction) [6]. They also investigated improved versions of the LSA model using word and document clustering. The model with word clustering used standard clustering techniques in conjunction with cosine similarity of the LSA semantic vectors for the words. The same method was used to cluster documents, but using the semantic vectors for each document instead. In language modeling, the word and document clustering approaches follow equations similar to topic modeling and traditional language models with word clustering. Under these models, not only is the probability of a word computed as a normalized cosine similarity (in this case, similarity to the cluster's centroid rather than the document history), but also the similarity of the cluster to the document is computed using normalized cosine similarity. The best perplexity results were obtained using 100 word clusters and 1 document cluster.[17] The word clustering approach on the bigram-LSA model reduced perplexity from 147 without clustering to 106 with clustering and document clustering reduced perplexity to 116. A model that uses both forms of clustering gave perplexity of 102 (overall 52.6% PP reduction from the original bigram model). The same trend is observed for the trigram-LSA model — word clustering reduced perplexity from 115 (baseline) to 98, document clustering gives a perplexity of 103, and the combination of the two gives a perplexity of 95 (overall 33.1% PP reduction from the original trigram baseline). Bellegarda [7] expands these experiments to include word error rate (WER) and a better representation of the document history. The document history is modified to decay exponentially using a weight. The optimal decay factor was found to be $\lambda = 0.975$, where a value of 1 signifies no decay and values close to zero gives very strong decay (the most recent word

[17]Intuitively, we would expect the results using a single document cluster to be identical to results without clustering, but no explanation is given for the difference between the two.

would dominate the scores). This method for exponential decay inspired our use of the technique. Using the direct adaptation mode (no clustering), they found that the bigram-LSA model reduced perplexity by 24.7% and WER by 13.7%. Using word clustering, the bigram WER reduction was increased to 22.5%. The trigram-LSA model with word clustering reduced WER by 15.8% over the baseline trigram model. To determine how portable the language model is, they also try repeating the same tests, but use a LSA model trained on AP news data from the same general time period as the testing data. They use the bigram-LSA model without clustering with the bigram component still trained on the original training data, but train the LSA model on between 44 million and 117 million words of AP news data. They find that the overall WER reduction ranged between 2.4% and 4.0% based on the amount of text used to train the LSA model, compared to 13.7% when the LSA component is trained on the original training data. The relation to our research is similar to Wandmacher and Antoine [83, 82]. The approach is difficult to iteratively update with user data due to the LSA method. In addition to the problem of updating the model, the approach only accounts for vocabulary usage in topic adaptations.

3.3.5.3 Trigger pairs

Research in trigger pairs identifies pairs of words where the first word increases the chance of encountering the second word later in the document [57, 59, 49, 65, 66]. For example, consider the probability of "car". We would expect the probability to be higher if we know that words such as "buy" and "auto" occur in the history. Pairs of words such as "auto" and "car" are called *trigger pairs*, where the first word triggers the occurrence of the second word. A special instance of trigger pairs is when both words are the same, called self-triggers. In this case, seeing a word once increases the chance that it will be seen again.

Li and Hirst [57] use a methodology similar to trigger pairs to utilize semantic knowledge in word prediction, but focus on words within the same sentence as the word to be predicted. Their approach seeks to combine two different ways of measuring related words. The first kind of knowledge for finding related words is the WordNet sense hierarchy and the second kind of knowledge is obtained from actual text, using pointwise mutual information (PMI), similar to earlier work in trigger pairs [49]. A semantic knowledge base is constructed for every noun in training (they use BNC, which is POS tagged). Given a noun w, they first measure the cooccurence of nouns in earlier parts of the sentence and adjectives in the past 5 words. Words with high PMI with w are retained as *seed words* and other words are added to a list of *candidate words*. The list of seed words is then expanded by adding any candidate words that occur in the WordNet glosses for the seed words. A relatedness score similar to PMI is stored for each noun and noun relative. In testing, the semantic association (SA) of a noun is measured as the sum of relatedness scores to the previous adjectives and nouns in the sentence. The SA score is then combined in a non-linear fashion with a baseline ngram model. They found that keystroke savings was increased from 59% to 65% for nouns using this approach. They varied the number of seed words as well as the size of the context for semantic association testing, and saved 0.94% keystroke savings by expanding the context from one sentence to three sentences (both two and four sentence contexts performed slightly worse). They also found that increasing the number of seed words from 10 to 30 improved keystroke savings by 0.24%. However, further expansions of the number seed words to 50 and 80 words both decreased keystroke savings. The process of adapting to the context is sometimes not useful because some documents may have many words which do not occur in training (OOVs). To address this problem, they design a simple unigram cache for named entities. When an uppercase letter is typed as the first letter of a word, they populate the predictions from the named entity cache before the full-fledged language model. The named entity cache is updated after every named entity is entered. This technique improved keystroke savings for nouns by an additional 8.53% over the semantic model. Li [56] provides additional details. The integration of the semantic component is implemented in a re-ranking paradigm, where the most likely 250 words are sent to the semantic component and rescored according to the combined ngram and semantic model. The parameter of 250 was chosen because they found that the semantic model was generally unable to boost the probability of words below 250th in the ranking to be in the top 10 words (the prediction list). They also describe an approach to help the semantic model in the absence of useful words in the short-distance context. They seek to identify keywords from the article that describe the topic of discourse. These *salient terms* are identified as words that have a frequency in BNC of under 15,000 (from 100 million) and a frequency in the article of 6 or more. Evaluation with and without this component was not performed, so it is unclear how beneficial salient terms are. While their goal is very similar to ours, their approach is very different. As with the LSA approaches, the topical generalizations made in training are computationally intensive, so user data is unlikely to be integrated back into the topic-based component. Also, the approach is limited to nouns and seems to blend the effects of selectional restrictions as well as the topic of discourse. It is unclear how to compare their findings on nouns to our findings on fringe words.

Matiasek et al. [60] outline the NLP research behind their multi-lingual word prediction system, FASTY. Much of the work is devoted to dealing with languages that are more inflected than English and also with strong compounding systems without spaces between compounds. They apply trigger words to word prediction in order to take advantage of the topical/semantic coherence of a document. In training, they identify trigger pairs like Lau et al. [49] by using average mutual information to rank candidate trigger pairs. Matiasek and Baroni [59] describe the trigger pair-based prediction component in more detail, and also evaluate the component for German. In training, trigger pairs are identified using pointwise mutual information (PMI). Cooccurences for PMI are measured in a sliding window through the training corpus. Words must occur at least 2 words apart but no more than 50 words apart to be considered a cooccurrence. Also, trigger pair identification is restricted to content words, which are identified using a morphological analysis tool. The PMI for each trigger pair is converted to a trigger score $S_t(w_1, w_2)$ by multiplying by an adjustable factor. In testing, the most recent n words are used for trigger scores. The scores from each individual word are combined using a linear combination, where the combination weights are based on the distance from the word being predicted. The previous word is given a weight of 1, the word before is given a weight of $\frac{1}{2}$, the word before a weight of $\frac{1}{3}$, and so on. The trigger-based scores are then converted to probabilities to be combined with the baseline language model. The overall model is a combination of the word unigram, word bigram, and POS tag trigram probabilities. The trigger-based probability is integrated by interpolating it with the word unigram probabilities. They trained the model on 27 million words of news text and tested on 11 news articles from a different corpus. Training found roughly 306,000 trigger pairs. They varied the percentage of the original word unigram weight given to the collocation-based module between 0% and 9% and found the highest keystroke savings at 7% of the original word unigram weight. This weight resulted in between 0.25% and 0.3% increase in keystroke savings. When testing on a larger test corpus with the same tuned parameters, they found that the trigger pair component increased keystroke savings by 0.178%.[18] They suggest that the method of combining trigger information with the standard model may be responsible for the modest improvement and that the maximum entropy model from Rosenfeld [66] might fare better. Trost et al. [79] present the whole FASTY system in implementation, including the trigger-based component, which they call collocation-based prediction. However, due to the moderate increase in keystroke savings of the component coupled with the difficulty of combining an extra language model, they omit collocation-based predictions from the production version of FASTY. They include a greatly simplified version of trigger pairs, focusing entirely on self-triggers (a word that is both the trigger and target). They maintain a user lexicon as the user types (even across documents), which contains word unigrams and bigrams for user text. These components are then interpolated with their general counterparts in the overall model. Initial testing shows that even with an initially empty user lexicon, this approach can improve keystroke savings by roughly 3%. Their approach is similar to our method of topic adaptation in spirit, but using trigger pairs. However, they found only a very small improvement in keystroke savings. We feel that the integration of the topic adaptations into the unigram component of the baseline model is to blame, similar to our findings with hybrid topic modeling. We address this limitation of adaptations by using pure topic modeling.

Gong [32] develops a word prediction method for text entry on cell phones and mobile devices using a combination of models based on frequency, syntax, and semantic relatedness. Like Tegic's T9$^{\text{TM}}$, a simplified language model is used to implement an ambiguous keypad. Gong's model is a linear interpolation of the three components, all of which are trained and tested on BNC text. The frequency component is a traditional unigram model and the syntactic model is the probability of the word in the most likely part of speech given the previous few words. The semantic relatedness component is a trigger pair model using word stems to reduce computational demands. The score relies on simple cooccurence probabilities, and is the sum of conditional probabilities for each word in the history. Unlike word prediction, they evaluate using disambiguation accuracy, which tells the percentage of the time the first suggestion is correct. When using a 3-buttion keypad, the syntactic and semantic components

[18]Although the improvement was small, statistical analysis showed that the change was highly significant.

raises disambiguation accuracy from 67.58% to 71.82%. In a user experiment, they found that the syntactic and semantic components improved words typed per minute from 7.01 wpm to 7.89 wpm. It is difficult to compare this work to ours as it is focused on not only a slightly different task (ambiguous keypad), but also is subject to very difficult computational and memory restrictions, thus limiting the possibilities for semantic adaptation.

Lau et al. [49] initially proposed the idea of a trigger pair for speech recognition where a trigger word A affects the probability of another word B, or triggers it. The three main problems in trigger models are 1) to identify trigger pairs in training 2) to combine multiple triggers and 3) to combine the evidence from triggers with a non-adaptive language model. They identify trigger pairs by scoring each pair of words with average mutual information. This measure is the weighted average of the pointwise mutual information of all combinations of each word's presence of absence with the other word's presence or absence. Lau et al. address the remaining two problems by using a Maximum Entropy (ME) framework, which is a machine learning method of the form $P(w) = \prod_i \mu_i^{f_i(w,h)}$ where w is the word whose probability is desired, h is the document history, the f_i's are binary functions, and the μ_i's are learned by the training algorithm. In the ME framework, each trigger pair is reformulated as a ME constraint that is 1 when the trigger pair is present and 0 otherwise. This method will create a constraint for each trigger pair. A unigram model is reformulated as ME constraints by having a constraint for each word in the unigram model which is 1 for the word and 0 otherwise. A bigram model is reformulated into constraints for each bigram that "fires" only when the bigram is present. They found that the ME framework with trigger pair, trigram, bigram, and unigram constraints reduced perplexity 12% over the baseline trigram model. One of the difficulties with this approach is that it is very computationally demanding, especially as further models are integrated. As part of the computational problems, integrating user text into the corpus may be difficult.

Rosenfeld [65] extended Lau et al. [49]'s work by building a larger scale combination model, which included a traditional trigram backoff model, a unigram cache model, a bigram cache model, and an extension of the previous ME framework. Their ME framework included trigger pairs and trigram, bigram, and unigram constraints as before, but also included distance-2 ngrams, which are ngrams with a gap. A distance-2 bigram, for example, checks that the word sequence matches $w_1 w^* w_2$ where w_1 and w_2 describe the actual bigram and w^* is allowed to match any word. A distance-2 trigram is similar, using the pattern $w_1 w_2 w^* w_3$. The baseline trigram model, ME component, unigram cache, and bigram cache were interpolated linearly. The weights for the interpolation were adjusted as more words were seen, reflecting the idea that the trigger model was more useful when more triggers had been encountered. They also varied the amount of training data used between 1 million, 5 million, and 38 million words of WSJ text. The methods were all evaluated on a held-out portion of 325 thousand words of WSJ text. They found that the ME model decreased perplexity over a standard trigram model, and moreso for the tests with less training data (ranging from 24% PP reduction at 1m words to 29% at 5m to 18% at 38m words). The interpolated model further reduced perplexity over the ME model (14% additional reduction at 1m, 9% at 5m, and 14% at 38m). In tests on separate testing data for speech recognition, they found that the system reduced word error rate by 10% (19.9% WER to 17.8% WER) when using the potentially error-laden decoder output for adaptation. They also studied domain variations by using the 38m WSJ model on AP news text and also comparing to a model trained on 5m words of AP news text. They found that both techniques reduced perplexity over the baseline in these tests, and that the interpolated model contributed more than the ME model for the cross-domain text (38m WSJ trained) and the ME model contributed more overall for the limited-data test (5m AP trained). As with other ME-based trigger pair approaches, it is unclear how to integrate stylistic adaptations and integrate user text back into training.

Rosenfeld [66] extends this with additional details and experiments with class-based trigger models. In a class-based trigger model, word clustering is used to group words into classes. Then trigger pairs are measured as (c_1, w_2) where c_1 is a class rather than a word. They used a stemmer to create word classes, but found that the results reduced perplexity by 2% in one case and increased perplexity by 0.2% with more training data. They also experiment with a linear combination of language models rather than using the ME framework and find that the ME framework is better for combining the models than linear interpolation.

Cai et al. [13] also applied the exponential language modeling technique to trigger pairs, but formulated the triggers differently. They instead sought to model semantic coherence within a sentence, rather than a document-level phenomena such as topic. They performed training for each potential word pair (w_1, w_2) by measuring the number of times the words co-occurred within a sentence, just w_1, just w_2, and neither. However, to avoid capturing the same relationships as trigrams, cooccurrences were only counted if the two words were 5 or more words apart. This table of cooccurence was used for Yule's Q statistic, which measures the association between the words on a scale from -1 to 1. They demonstrated the inadequacy of trigram models for this task by comparing Q statistics on natural sentences as well as sentences generated using a trigram model. Their evaluation showed 1.5% and 3.5% reduction in perplexity over the baseline trigram model using two different methods for integrating the semantic coherence triggers. Their variations on previous approaches to trigger pairs does not address the problems for applying trigger pairs to word prediction.

3.3.5.4 Other topic adaptations Other researchers such as Berger and Miller [8] and Lesher and Higginbotham [53] have adapted a language model to the topic of conversation by periodically using web searches to augment the training data with additional on-topic data, using keywords from the current document as the query. These approaches are very similar in spirit to the information retrieval style of Mahajan et al. [58], but using a dynamic, online corpus from which to draw topic adaptations, rather than a static corpus with a researcher-created similarity score. This approach can potentially adapt to the topic of discourse as well as increase the training data, however, AAC devices may not have a reliable Internet connection to use this approach.

4 Proposed Further Work: Adapting to Style and Cache

4.1 Style adaptation

Style is the aspect of language that is content-free, even though certain styles may be correlated with particular topics through a sublanguage (e.g., legalese). One of the most clear-cut stylistic differences is between spoken and written text [10] (e.g., more pronouns and deictics in speech, longer sentences and words in writing). We believe style is particularly relevant for AAC devices because a single device is used to generate language in a variety of styles (e.g., conversation, email, pre-planned speech, articles). This is in contrast to traditional word processing, which is generally used for a more formal style of language. Consider the following example of similar content written in two different styles, from our corpus study in Section 4.1.2:

> Switchboard is really low. This could reflect that we chose a good corpus originally, maybe that the cleanup was more consistent (I don't think it's any more advanced than the others, but I think I spent far more time on it) Another possibility is that since subjects were restricted to talking about predefined topics, there's enough data to cover the words for those topics well.
>
> Switchboard shows the lowest percentage of OOVs in the self-test, whereas the larger Slate corpus shows a much higher amount of OOVs. The self-test analysis is affected by both the size of the corpus as well as the diversity of the corpus, which explains the trend with Switchboard: participants in the corpus collection were restricted to one of roughly 70 topics, most of which are represented in every set of Switchboard.

Although stylistic differences are relatively easy for a human to identify, it is unclear how to programmatically identify whether these two texts are from similar or different styles, let alone design a language model that accounts for differences in style. Therefore, we will examine style in applied research in order to more concretely define style in Section 4.1.1, focusing on work in genre classification and Hovy's work in generation [37]. The review of existing research will particularly focus on style as a global phenomenon, affecting the entire text, in the manner that speaking or writing affects an entire text. After examining related work, we will present a case study that computationally identifies some of the differences in style between two corpora in Section 4.1.2. Finally, we will present our plans for creating a style-adapted language model in Section 4.1.3 based on our findings with related work, our corpus study of style, and prior findings with topic modeling.

4.1.1 Related work

Hovy [37] modeled style for the goal-driven generation system PAULINE. The high-level input to the system included the available semantic knowledge, the conversational setting, and the interpersonal goals. These high-level inputs were broken down into rhetorical goals in the model, which more closely fit the traditional view of style, including goals such as formality, simplicity, detail, haste, force, respect, personal reference, etc. An example of the processing in PAULINE is that the conversational setting "time" would directly affect the rhetorical goal "haste". The conversational setting regarding the speaker-hearer relationship would affect the "respect" rhetorical goal. Interpersonal goals such as informing the hearer or affecting the speaker-hearer relationship affect rhetorical goals in a similar manner.

Each of Hovy's rhetorical goals (which are aspects of style) has a range of values. For example, formality ranges from "highfalutin" to "normal" to "colloquial". Floridity ranges from "dry" to "neutral" to "flowery". The settings of each rhetorical goal can slant the generated text in two general ways — either in content selection or form-related, which affects the ordering of sentence parts, inclusion of enhancers and mitigators to promote/demote concepts, and selection of the appropriate word amongst alternatives (e.g., choosing between "thrifty", "frugal", "cheap", and "stingy"). Some of the rhetorical goals are achieved using an agreement process — words are annotated with the values and then matched with the goals' values. For example, "bug" might be labeled as colloquial, whereas "insect" might be labeled as highfalutin. If formality is colloquial, then "bug" would be selected over "insect". A second example of lexicalized formality is "integrate" (highfalutin) compared to "combine" (neutral). In contrast to previous goals, some goals can affect the structure of a sentence. For example, Hovy says that formal texts tend to have more conjunctions and multi-predicate phrases, as well as increased use of passive voice and lack of reference to the speaker/hearer. On the other hand, he finds that informal texts tend to have more pronouns and use simplified tenses as well as less redundant modifiers.

DiMarco and Hirst [24] also acknowledge that a computational treatment of style is desirable, particularly for machine translation and other research involving generation. However, their treatment of style deals with very small units of text, such as a single sentence, unlike the way in which we seek to account for style. Similarly, Green [34] focuses on very local phenomena for style, primarily ellipsis, but also seeks computational treatment of style due to its potential. In other work, DiMarco et al. [23] focus on lexicalized style in near-synonyms for machine translation, attempting to preserve the tone of translated text.

In the domain of language modeling, work on style appears to be scarce except when sublanguage is implicitly modeled (without any explicit linguistic phenomena being modeled). Some research in adaptive modeling uses the EM algorithm as the basis for adapting a linear combination [18, 31, 38, 1]. This adaptation is based on the probability that each individual topic language model assigns to the part of the document seen so far, tuning the weights such that the combination results in the greatest likelihood for the document seen so far. These automatic optimization methods model both topic and style, as they adapt both to the vocabulary as well as grammar and the way words flow from one to the next. Research in topic modeling that uses only word unigrams to identify topics may implicitly model style to some extent, because there tend to be correlations between topic and style. Conversations about sports tend to be informal, for example. Although most researchers in topic modeling are more interested in the adaptive or topical aspects of the field, Clarkson and Robinson [19] seek to use the EM algorithm to adapt a language model to both topic and style. However, they model both topic and style by using a similarity score that can identify both, whereas we plan to have separate adaptations for topic and style.

Additionally, efforts have been taken to automatically detect text genre for Information Retrieval [51, 22, 61, 72, 73, 44, 42, 62, 64, 86]. The unifying motivation in this work is to classify documents by genre (e.g., news, blog, corporate, sale, etc.) in order to make it easier for a user to find the type of document they seek in addition to finding the correct content. For example, a student writing a report may only want to retrieve factual web pages, such as news and peer reviewed publications. On the other hand, a consumer searching for new headphones would prefer product reviews and listings. Karlgren and Cutting [42] study genre classification on the Brown corpus, using the corpus-supplied hierarchy. They use various counts (e.g., character count, adverb count, long word count, second person pronoun count) as features for discriminant analysis and find that they can classify by genre fairly well — 96% accurate for 2 classes, 73% accurate for 4 classes, and 52% for 10-15 classes. Kessler et al. [44] also study genre classification on the Brown corpus, but create their own classification scheme due to concerns about the existing categories. They view style as a three-level process. The most general level is genre, such as reportage, editorial, legal, etc. Each genre can be described using attributes called facets, such as narrative (yes/no) or brow (popular/middle/upper-middle/high). Each of these facets is recognized using cues, which can be at various linguistic levels. Kessler et al. separate cues into structural (frequencies of syntactic phenomena, including POS counts), lexical (counts of specific words), and character-level cues (mostly punctuation). Derivative cues are combinations of individual cues. Logistic Regression and two Neural Networks are trained on the lexical cues. The classification performance is compared against a predict-majority algorithm as well as the structural features of Karlgren and Cutting [42] for classifying each facet. Kessler et al. find that they can classify significantly better than the baseline (predict majority) and are often comparable to Karlgren and Cutting's structural features. They conclude that genre can be classified just as well without structural information. Michos et al. [61] discusses a layered view of style more formally. Their most general layer is called categories, which are the basic categories of style such as public affairs, scientific, journalistic, etc. The second level is called main features, which are similar to Kessler et al.'s facets. Example features are formality, elegance, syntactic complexity, and verbal[19] complexity. The most specific level is linguistic identifiers, which are further broken down into verbal and syntactic identifiers. Example verbal identifiers include idiomatic expressions, scientific expressions, and abbreviations, whereas syntactic identifiers include sentence length, verb-noun ratio, active-passive ratio, etc. They give hand-crafted rules that describe how linguistic identifiers identify each feature, and in turn how each feature can identify each category of style. Michos et al. [62] apply this notion to style classification in further work. Each linguistic identifier is a numerical score, often a count or a ratio. They compute the average counts and ratios for their overall corpus and then look at an individual text, highlighting whether the counts and ratios are higher, lower, or about the same. For each feature (e.g., formality), they use the hand-crafted rules to see if the identifier agrees with the description, disagrees, or neither. They then decide whether each feature is present to a large or small extent, or not able to be determined. The features present to large and small extents are then matched with the hand-crafted descriptions of style categories. They present this process in detail as a case study of one document and then show that it recognizes style fairly well for some other documents. Stamatatos et al. [73] expand on this work by using machine learning rather than hand-crafted rules. They also replace manual evaluation with automated analysis. They apply this method to genre and author classification. They find that genre can be classified with 82% accuracy and author can be classified with about 70% accuracy. There were 10 classes for each task and the same number of documents in each class, therefore a predict-majority method would be 10% accurate. Interestingly, they perform analysis by text length, showing that most classification errors occur on small documents. This result makes sense, as smaller documents are poor samples of each class. There have been a few other approaches to fully automated genre classification. Wolters and Kirsten [86] studied topic and genre classification for German news text, using the distributional similarity of unigram distributions of part-of-speech for classification. Their results are promising. Dewdney et al. [22] experiment in genre classification using many features as well as several classifiers. They have word features, which are pruned using information gain, and many presentation features, including sentence length, word length, basic POS, punctuation, etc. They tried Naïve Bayes, C4.5 decision trees, and SVM-light for learning algorithms and found that both the word-based and presentation features contributed overall by experimenting with both features sets together and apart. They found that Naïve Bayes was more suited to word-based features and C4.5 and SVM-light were more suited to presentation features. They found that genre could be classified with 92% average recall and using a threshold, 95% precision and 84% recall.

Although genre classification has its roots in very syntactic features (e.g., verb-noun ratio, active-passive voice ratio), researchers have moved towards classification with more simple features, such as just words. Statamatos et al. [72] use the most frequent 100 English words along with punctuation to classify by genre, using the frequency of each word and punctuation mark as a feature for discriminant analysis. They find that BNC is a better corpus

[19] In their work, they seem to use the term "verbal" to mean "lexical".

for determining the most frequent words than the training corpus (WSJ) and that performance peaks at roughly the 30 most frequent words and lessens with more. They find that punctuation marks improve performance over just word counts. They found that the combination of both types of features could bring accuracy over 97%. However, a predict-majority baseline or equivalent was not supplied, so the difficulty of the classification task is unknown. Lee and Myaeng [51] exclusively use word unigrams as the features in genre classification. They experiment using cosine similarity and Naïve Bayes and additionally test a method like IDF for genre classification. They find that the similarity-based classification using cosine is better and that the IDF-like weighting improved performance. They found a maximum of 87% micro-averaged precision/recall for English and 90% for Korean.

Peng et al. [64] take the shallow information a step further and use only *character ngram models* for multiple classification tasks. One of the motivations behind character ngram models is the difficulty of word segmentation for Chinese and Japanese. Rather than build a classifier on top of potentially erroneous segmentation, they prefer to use the characters directly. Given a set of classifications, they compute character ngram models for varying n. When presented with a new document, they apply a backoff ngram model from each category and select the category that maximizes the probability of the document. They varied the smoothing method used for backoff among absolute, Good-Turing, linear, and Witten-Bell and varied n from 1–9 depending on the task. They study language identification (6 languages), Greek authorship attribution, Greek genre classification, English topic detection, Chinese topic detection, and Japanese topic detection. They find performance better than earlier work in the Greek tasks and English topic detection, and comparable to SVM-based methods for Chinese and Japanese topic detection. Bigrams were best for some tasks and 6-grams for other tasks.

Our intuitions along with cited research demonstrates that style is useful for some tasks and is a tangible linguistic phenomenon. We plan to model style as a two-step process of identification and then adaptation. Research in genre classification may help with the identification step, as the tasks are very similar. However, one criticism of genre classification research is that some methods allow infrequent words to be used in classifying genre. It is possible that such approaches may be actually classifying by topic if the topic(s) of each genre are different. For example, if reportage and reviews are two genres as in [42], words about books and movies are likely to be useful in classifying reviews, but those words are topic-related, not style-related. Although this is not a significant problem for genre classification, it is a problem for our approach to language modeling because our model of style will be combined with a model of topic. If our model of style were based on topic in part, it would be unlikely to improve word prediction beyond the topic model. However, if our style model handles style alone, it presents an opportunity to combine two complementary sets of constraints: topic and style. Therefore, we will focus on those features that are clearly not related to topic, such as grammatical information.

4.1.2 Corpus study

In addition to examples of style and established research, we seek to observe stylistic differences empirically. To accomplish this, we constructed two corpora of approximately the same topic but different styles: one collection of our research emails and another collection of our technical papers. We compared the following distributions between the two corpora:

- word unigrams
- WSJ part-of-speech (POS) unigrams
- WSJ part-of-speech (POS) bigrams
- classed WSJ[20] part-of-speech (POS) unigrams
- classed WSJ part-of-speech (POS) bigrams

[20]The WSJ POS tags were placed into more general classes, shown in Appendix B. For example, all variations of pronouns and different nouns were grouped together under the general noun tag and likewise for verbs, adjectives, and adverbs.

4.1.2.1 Style-varied Texts We collected 33 emails containing about 8,800 words involving communication between our research group regarding the fringe word prediction project. This corpus represents a somewhat informal style of writing. The formal style corpus contains 4 papers with about 15,500 words intended for publication. The following is a sample of writing from the email corpus:

> Switchboard is really low. This could reflect that we chose a good corpus originally, maybe that the cleanup was more consistent (I don't think it's any more advanced than the others, but I think I spent far more time on it) Another possibility is that since subjects were restricted to talking about predefined topics, there's enough data to cover the words for those topics well.

The following is a sample from the papers collection:

> Switchboard shows the lowest percentage of OOVs in the self-test, whereas the larger Slate corpus shows a much higher amount of OOVs. The self-test analysis is affected by both the size of the corpus as well as the diversity of the corpus, which explains the trend with Switchboard: participants in the corpus collection were restricted to one of roughly 70 topics, most of which are represented in every set of Switchboard.

The two corpora were preprocessed to remove any artificial line breaks due to word wrap and then part-of-speech tagged using Kristina Toutanova's tagger[21]. For more coarse-grained POS classes, we used the mapping of POS tags shown in Appendix B.

4.1.2.2 Methods The corpora were part-of-speech tagged and then processed in Perl to compute the different distributions (word unigram, POS unigram, POS bigram, etc.). The two corpora were compared for each distribution. For example, a small portion of the word unigram comparison is shown in Figure 18. The frequency (f) and probability (p) of each word in each corpus are shown. Additionally, we measured the ratio between the probabilities (p1/p2), so in the above example, "I" is 16.6 times more likely in emails than papers and "of" is 2.3 times more likely in papers than emails. We also computed a difference score (d) to judge how much the word varied in probability between the two corpora. The difference score is measured by taking the difference in probability over the average, shown below:

$$d = \frac{\mid P_{email}(w) - P_{papers}(w) \mid}{\frac{P_{email}(w)+P_{papers}(w)}{2}}$$

The difference score ranges from 0 to 2. Words with $d \geq .5$ were deemed to be different for the purposes of analysis. We mapped the brightness of table cells to the value of d in order to ease interpretation.

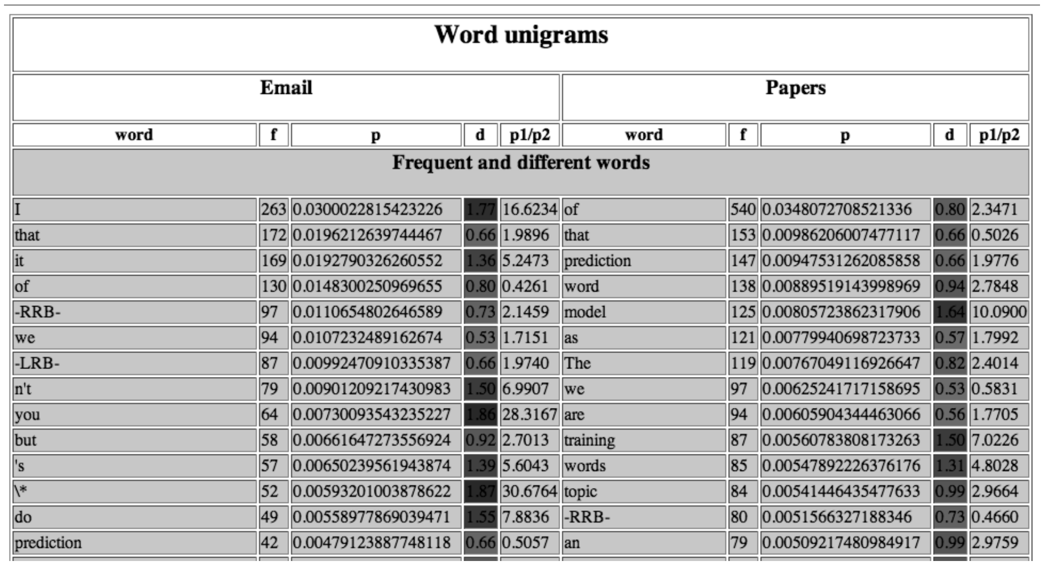
Our analysis tool presented the distributions in three segments:

1. words which occur frequently ($f \geq 3$) in both corpora, but occur with very different probabilities ($d \geq 0.5$)
2. words which occur frequently ($f \geq 3$) but have similar probabilities ($d < 0.5$)
3. words which occur infrequently ($f < 3$) in either corpus

Infrequent words were not separated by whether they differed, as we felt that analysis based on infrequent words was unreliable.

We focused our analysis on words that were both different and frequent. When inspecting words in emails, for example, we focused on the first region (frequent and different words) and focused on those words that were very frequent and/or very different. Words that were very frequent and different in a corpus were considered to be

[21]Downloaded from `http://nlp.stanford.edu/software/tagger.shtml` around 6/20/2007



Word unigrams

Email					Papers				
word	**f**	**p**	**d**	**p1/p2**	**word**	**f**	**p**	**d**	**p1/p2**
Frequent and different words									
I	263	0.0300022815423226	1.77	16.6234	of	540	0.0348072708521336	0.80	2.3471
that	172	0.0196212639744467	0.66	1.9896	that	153	0.00986206007477117	0.66	0.5026
it	169	0.0192790326260552	1.36	5.2473	prediction	147	0.00947531262085858	0.66	1.9776
of	130	0.0148300250969655	0.80	0.4261	word	138	0.00889519143998969	0.94	2.7848
-RRB-	97	0.0110654802646589	0.73	2.1459	model	125	0.00805723862317906	1.64	10.0900
we	94	0.0107232489162674	0.53	1.7151	as	121	0.00779940698723733	0.57	1.7992
-LRB-	87	0.00992470910335387	0.66	1.9740	The	119	0.00767049116926647	0.82	2.4014
n't	79	0.00901209217430983	1.50	6.9907	we	97	0.00625241717158695	0.53	0.5831
you	64	0.00730093543235227	1.86	28.3167	are	94	0.00605904344463066	0.56	1.7705
but	58	0.00661647273556924	0.92	2.7013	training	87	0.00560783808173263	1.50	7.0226
's	57	0.00650239561943874	1.39	5.6043	words	85	0.00547892226376176	1.31	4.8028
*	52	0.00593201003878622	1.87	30.6764	topic	84	0.00541446435477633	0.99	2.9664
do	49	0.00558977869039471	1.55	7.8836	-RRB-	80	0.0051566327188346	0.73	0.4660
prediction	42	0.00479123887748118	0.66	0.5057	an	79	0.00509217480984917	0.99	2.9759

Figure 18: Screenshot of comparison between email word unigrams (left) and paper word unigrams (right)

typical examples of the specialization of that corpus. For example, "I" is one of the words that is much more characteristic of emails. Because it is also very frequent in emails, we know that it is a more reliable characteristic of emails than less frequent words. We will focus primarily on these typical examples when discussing each analysis.

We also performed this analysis for POS unigrams and bigrams in addition to words, treating a POS tag as a word for POS unigrams and a POS tag pair as a word for POS bigrams. The two POS analyses were run using both the standard Treebank tags as well as more general tags. In the total four POS analyses, we added a tool to allow quick interpretation of data by showing the 10 most frequent instances of the POS tag or POS tag bigram in a tooltip. The complete set of analyses is available online at `http://www.cis.udel.edu/~trnka/work/style/`.

4.1.2.3 Results

Each distribution will be presented separately with typical examples from each corpus' distribution, focusing on examples that differentiate each corpus.

Many of the characteristic words[22] of emails and papers are shown in Table 15. Each token is presented with the ratio of the in-domain probability over the probability from the other corpus. The probability ratio is shown in this table as it intuitively describes the higher likelihood of a word in each corpus. For example, "you" is 28 times more likely in emails than papers and "model" is 10 times more likely in papers than emails.

Originally, we expected that a comparison of the word unigram distributions of two different styles of text would show that predominantly closed-class words, such as discourse markers ("however", "so", etc.) and pronouns, would comprise the majority of the differences. The analysis shows a difference in closed-class words, such as the higher likelihood of pronouns (typical of informal text) and contractions in papers.[23]

However, word unigram analysis also shows several open-class words differing between the corpora. The higher likelihood of "paper" in emails than papers is due to the slightly different topic of emails. Our research emails often discuss writing papers, whereas papers focus purely on the research. The presented characteristics of papers are all content words. However, the topic of language modeling and keystroke savings is often discussed in emails as well as papers. The reason these content words are showing up is that the audience for emails doesn't require

[22]Words that are both very frequent and very different

[23]The POS tagger we used tokenized "don't" as "do" and "n't", so usage of contractions shows up differently in the distributions than normal words.

any background knowledge, whereas we present the background of our research in papers, as the audience is often unfamiliar with our work.

Emails	
Word	**Probability ratio**
you	28.31
I	16.6
paper	14.2
things	13.6
n't	6.99
's	5.6
it	5.24
if	4.4

Papers	
Word	**Probability ratio**
model	10.1
language	8.36
models	7.72
training	7.01
keystroke	6.50
savings	6.36

Table 15: Word unigrams

We compared the unigram distributions of part-of-speech (POS) tags between the two corpora to investigate any grammatical differences. Table 16 shows the significant differences in the POS tags of each corpus. As with word unigrams, POS unigrams show more characteristics of emails than papers (i.e., there are many POS tags that are more likely in emails, but few such tags in papers). The trend with pronouns that was shown in word unigrams in emails is also pronounced in POS unigrams. There are some interesting trends with verbs — base forms, non-3rd person present, and past tense conjugations are more likely in emails whereas past participles are more likely in papers. Bracketing is much more likely in emails and symbols are similar — it is more common in email to have code such as "svn add *.tex". The much higher frequency of particle verbs in email is interesting — a quick look over the particles shows terms like "write up", "figure out", "nail down", "mock up", and "set back". Intuitively, these verbs seem unlikely candidates for formal writing. The higher proportion of comparative adverbs in scientific papers is likely due to the content matter — where it is important to evaluate and compare multiple approaches. Formal papers are much more likely to use the passive voice and therefore we find a much higher proportion of past participle verbs in papers. In addition, foreign words (e.g., Latin) are more likely in papers, typical of formal academic writing. Grouping the parts of speech into more general classes (shown in Appendix B) did not reveal new information. Nouns, verbs, adjectives, and adverbs were generally similar between the two styles.

Emails	
POS	**Prob. ratio**
SYM (symbol)	28.9
RP (particle)	8.17
PRP (personal pronoun)	4.56
WP (Wh-pronoun)	3.54
EX (existential "there")	3.21
-RRB- (right paren/bracket)	2.16
-LRB- (left paren/bracket)	2.00
VBD (verb, past tense)	1.82
VB (verb, base form)	1.78
RB (adverb)	1.71
VBP (verb, non-3rd sing. pres.)	1.67

Papers	
POS	**Prob. ratio**
RBR (adverb, comparative)	1.99
VBN (verb, past participle)	1.88
FW (foreign word)	1.75

Table 16: Part of speech unigrams

The higher likelihood of pronouns in emails obscured the results of analysis on bigrams; the majority of the characteristic POS bigrams of emails included a pronoun. Therefore, we used the coarse-grained POS tags for bigram analysis to get a better feel for the bigram differences in style, independent of the specific differences in the unigram POS results. Some of the notable features of each style are shown in Table 17. Papers seem to have more modified nouns than email — both noun-noun[24] and adjective-noun[25] pairs are more likely in papers. There were also more strings of multiple adjectives in papers[26]. As noted before, phrasal verbs are more common in papers. This was seen in the high number of particles with POS unigrams, and is now shown with POS bigrams with the `V PART` bigram 6.75 times more likely in email. Common phrasal verbs in emails include "write up" and "figure out". Due to the tokenization of "don't" into two tokens, `n't/ADV` affected the bigram analysis somewhat: the ADV-V and V-ADV were much more likely in emails due to contractions. Among the N-ADV bigrams are "I just", "I also", and "I still". Among the ADV-N bigrams are "so I", "so you", "then I", and "not something".

Emails	
POS pair	**Probability ratio**
WP N	9.29
V PART	6.75
ADV N	3.79
WRB N	2.94
CONJ ADV	2.34
MD ADV	1.94
ADV ADV	1.91
. N	1.84
TO V	1.79
V ADV	1.77
N ADV	1.75
V TO	1.73
ADV V	1.72

Papers	
POS pair	**Probability ratio**
. P	3.23
. DET	2.71
CONJ ADJ	2.70
P ADJ	2.17
ADJ ADJ	1.96
N N	1.82
ADJ N	1.74

Table 17: Coarse-grained part of speech bigrams

4.1.3 Modeling style adaptations

We have demonstrated that style can be recognized computationally by comparing distributions of part of speech tags and pairs between texts. Although other types of features have been used for style classification, the discriminative power of features such as punctuation, character ngram models, and sentence length are diminished greatly when dealing with a collection of independent corpora, where these features may not be consistent across corpora or even present in all corpora.

A style-adapted language model will tune word prediction to conversational needs (e.g., formality, time constraints), complementing a topic model which focuses on adapting to the content of conversation. Our experience with adapting to topic leads us to believe that a similar process may be applicable to style modeling — splitting the component into style identification (i.e., what is the current style?) and style application (i.e., how should style affect the overall language model?). We will focus on identifying style using part of speech tags and pairs and build a Markov model that allows style identification to affect transition probabilities in a part of speech ngram model. In language modeling, this builds on existing research in class-based ngram models [12], which group words into classes that occur in similar contexts. While Brown et al. [12] focus on automatic word clustering to improve language modeling, a similar model is often applied to part of speech (POS) tagging). The basis of a

[24] "word prediction" and "keystroke savings" were the two most frequent in papers.
[25] "basic prediction", "in-domain training" and "out-of-domain training" were the 3 most frequent in papers
[26] Examples include "several conversational", "significant cognitive", "much larger", and "most common"

Markov model POS tagger is the usage of a probability function in conjunction with an optimization algorithm (see [41, 40] for further details):

$$P(w \mid h) = \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}, \ldots) * P(w \mid tag)$$

where tag is the hypothesized POS tag, tag_{-1} is the hypothesized tag of the previous word (determined by the Viterbi algorithm), and $POS(w)$ is the set of possible tags for word w. The Viterbi algorithm is a dynamic programming solution to an exhaustive recursive search over all POS tag assignments, which searches for the most probable sequence of tags using the above equation. In general, the emission or posterior probability $P(w \mid tag)$ is computed based on the frequency $f(w \mid tag)$. In the case of a trigram Markov assumption, the transition or prior probability $P(tag \mid tag_{-1}, tag_{-2}, \ldots)$ is computed based on the frequency $f(tag \mid tag_{-1}, tag_{-2})$. When the sparseness of this distribution is problematic, it may be computed as an interpolation with lower-order distributions, as in Kuhn and de Mori [47].

We seek to extend the POS ngram model above by conditioning the prior on the style:

$$P_{style}(w \mid h) = \sum_{s \in styles} P(s \mid h) * \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}, s) * P(w \mid tag) \tag{5}$$

where s is the hypothesized style. This approach is the application of our topic model to style modeling, but using a POS ngram model rather than a word ngram model, and only conditioning the prior on the style. We hypothesize that the style will have a substantial effect on the POS tags, however, we do not expect the style to have much affect on the specific words used. We feel that the specific words used will be more affected by topic than style. Therefore, we only condition the POS prior on the style and not the posterior probability.

Firstly, we will compute $P(w \mid tag)$ based on frequencies in the training data. We plan to initially compute $P(tag \mid tag_{-1}, tag_{-2}, s)$ by measuring frequencies in text of style s. If data sparseness is an issue with our corpora, we will investigate dropping the component $tag - -2$ as well as computing it as a linear interpolation of frequency-based estimates from lower-order models. We plan to model the style probability $P(s \mid h)$ by using relevance scores, which were successful for topic modeling. The remainder of this section will outline our plans to address the remaining problems in style modeling, such as more specific computation of style relevance scores and how we will obtain a collection of texts segmented by style.

The style identification score will be computed using similarity of POS tag and pair distributions. We found that POS unigrams and bigrams showed substantial differences between texts of similar topic but different styles. Additionally, many researchers have used POS tags for genre classification [42, 44, 86, 22], a somewhat similar problem to style modeling. Other research in genre classification uses POS tags combined with some processing (e.g., complex syntax, active-passive voice ratio) [62, 61, 73]. Because POS tags are our choice for what the language model will actually represent, they are also a natural choice to apply in style identification. One of the other advantages of using POS tag distributions for style identification is that we can draw on our experience in distributional similarity for topic identification. We plan to apply the cosine measure with inverse style frequency (ISF), building on our findings with topic modeling. The one major difference with style is that we want to apply similarity of POS bigrams in addition to POS unigrams. We will approach this problem first by trying to lump the unigram and bigram distributions together and letting the ISF measure focus the similarity score on the relevant features of the two distributions. If that approach fails, we will try applying Naïve Bayes to the problem, which may combine the two sources of information better, or else compute cosine similarity of the two distributions separately and then take a weighted average of the two scores. In summary, we plan to use POS tags in both the modeling part as well as the identification part, and we will build on our experiences in topic modeling for style identification.

There are other potential features from genre classification that we could apply to style identification. Many researchers use punctuation, however, punctuation is inconsistent in spoken corpora, generally reflecting the annotator's choices rather than the speaker's language. Therefore, we don't plan to use punctuation for style identification. Average word and sentence length are also fairly common features which seem to help in genre classification (and are certainly complementary to topic/vocabulary). If style identification with POS alone proves difficult, they may be useful features. Some researchers have used word unigrams in style classification. However, this raises questions as to whether such measures are only capturing style or topic. We feel that they capture a mix of both, and are successful in genre classification because corpora broken down by style are not homogeneous with respect to topic. A better way to use word distributions is to look at the frequent words only, essentially applying the opposite of the IDF measure. While we feel that this method has some potential, we feel that it would be best suited for adapting the frequent vocabulary, whereas we are focusing on fringe vocabulary. As with word and sentence length, we will initially discard this feature, but may revisit it if POS tags prove to be insufficient for style identification.

The main remaining problem to address in style modeling is defining what the set of styles will look like. As with topic modeling, our approach requires a predetermined set of styles. Our experiments in topic granularity provide some insights into this problem. The first option is to use an approach like fine-grained topic modeling and allow the style identification process to weight each document appropriately. Like fine-grained topic modeling, the main disadvantage of this approach is computational. A second disadvantage is that fine-grained topic modeling doesn't seem to perform as well as medium-grained topic modeling. However, fine-grained modeling avoids the need for corpus annotation. The second option is to use an approach like coarse-grained topic modeling, where each corpus is treated as a style. This assumes that corpora are relatively homogeneous in style. In the case of the major features of style (e.g., spoken/written distinctions, technical/general), we expect that corpora are good candidates for sets of styles. Our collection of corpora meets this requirement, where each corpus contains either spoken or written language, but not both. Many of our spoken corpora are restricted to a specific conversational setting (e.g., telephone conversations between strangers, phone calls to friends/family, transcribed face-to-face interviews). Hovy [37] used conversational settings such as this to select the appropriate style in generation, suggesting that our corpora may have consistent styles. Although our corpora are relatively homogeneous with respect to general stylistic features (spoken/written, face-to-face, etc.), they may be heterogeneous with respect to more specific features (e.g., politeness, syntactic complexity). The final option is to use the style identification measure as the similarity metric in automatic clustering. This approach addresses the computational demands of fine-grained modeling and may also address the problems of coarse-grained modeling. We will initially take each corpus as a style for simplicity.

One criticism of our approach is that lexicalized style (e.g., choosing "however" vs. "but") will not be modeled. We feel that lexicalized style is already modeled to a small extent using the topic modeling method; for example, words with high IDF that have formal and informal variants should affect topic identification and thus affect the predictions. However, we suspect that in general lexicalized style isn't modeled well by the topic modeling component. Although adapting the vocabulary to style may be beneficial in the long term, we will focus on POS as our starting point in style modeling. If we find that style is not modeled well, then we will revisit the question of modeling lexicalized style.

One of the major hurdles remaining for style modeling is how to deal with probabilities, frequencies, and smoothing. The addition of a second linear interpolation complicates the problem. The first subproblem is finding a way to model style that avoids data sparseness problems. Sparseness is less of an issue with a POS model due to the smaller parameter space (although the most constrained component uses three conditions, there are significantly fewer POS tags than words). It may be possible to perform smoothing and backoff to calculate $P(t \mid t_{-1}, t_{-2}, s)$ if medium-grained or coarse-grained style modeling is used. Alternatively, if data sparseness is an issue for the POS tag transition probability, we may be able to rewrite the equation to interpolate frequencies for the transition probability and then smooth them afterwards for a backoff model. The second subproblem is computational in nature; topic modeling was made practical because word probabilities need only be computed for the 10 most likely trigram, bigram, and unigram candidates, limiting computation to 30 words at most. In the case of the style-adapted POS ngram model, it seems that iteration over nearly all possible POS tags is necessary. However, the linear combination presents a new dilemma for optimization beyond a backoff model. In a true backoff model, the probability of a word comes from only one of the submodels. Thus, we can guarantee the top 10 words to be in the set of the combined top 10 lists from each individual model. In a linear combination model, however, it is possible that candidate #11 in every model is the same. This candidate would likely rank higher overall than a different word that is candidate #10 in one model and candidate #99 in all other models, but if we used the backoff model optimization on this linear interpolation, we have a lossy runtime optimization; keystroke savings would decrease (possibly by a very small amount, but it would decrease nevertheless) for the benefit of faster runtime. However, many words have a predominant part of speech, as demonstrated by Charniak et al.'s findings with a very basic POS tagger [15]. Words that occur in multiple parts of speech will probably be much more likely in one particular POS in a given context. So for the most part, we feel that a 10-best list for each possible POS tag should affect the keystroke savings in a negligible manner.

4.2 Cache-based adaptation

We seek to fine-tune our training to the topic and style of conversation to account for the mismatch between the topic and style of training and testing data. We can also address this problem by iteratively training a language model on the testing data after it has been processed, called a cache model. The fundamental difference is that topic and style models do not expand the training data from the original training set, but merely re-focus training on the most relevant portions. On the other hand, cache models seek to expand the training data by using testing data for re-training after it has been encountered in testing.

Jelinek et al. [39] explored both unigram and trigram cache models and interpolated them with a trigram model from training data for speech recognition. The purpose of their adaptations were to match the language model to the testing document more closely, like our goals to compensate for the differences between training and testing data, except that their training and testing data were both from the insurance domain. They measured the cache models on the most recent 1000 words of testing data (and did not reset the cache between documents). The baseline trigram model gave a perplexity of 262 on testing data, compared to 230 with the unigram cache interpolated with the baseline and 190 with the trigram cache interpolated with the baseline after tuning interpolation weights. Wandmacher and Antione [83] also experimented with unigram and trigram cache models, but applied the models to word prediction using very different training and testing corpora. They found that the unigram cache model improved keystroke savings somewhat for all of the testing corpora using a decay factor to take recency into account (0.47% – 3.53%). The trigram cache model, or dynamic user model, improved keystroke savings much more (6.6% – 14.64%) However, unlike Jelinek et al.'s model, they never reset the trigram cache. Because they test separately on different corpora, the trigram cache model will become similar to a model created with training data from the testing corpus. Although the evaluation is valid, we feel that actual AAC usage may alternate between very different styles of text, which would diminish the benefit of including all past testing data in the cache model. Wandmacher and Antione also explore the usage of a unigram model purely for out-of-vocabulary words (OOVs). When an unknown word is encountered in testing, it is added to the user dictionary. Then when a new word is predicted, a very small amount of probability is given to the user dictionary model. This approach improved keystroke savings similarly to the unigram cache model with recency, but Wandmacher and Antione found that it reduced the percentage of OOVs in testing by 1.22% – 6.99%. Li and Hirst [57] also explored the use of cache modeling to deal with OOVs in word prediction. In contrast to previous research, however, they focus only on named entities, creating a unigram cache of uppercase words. In testing, if the user begins the word with an uppercase letter, the prediction window is filled from the named entity cache first, and if the cache does not provide enough predictions, the baseline method provides the remaining predictions. They found that the named entity cache increased keystroke savings by 8.5% on nouns. Their keystroke savings is measured only over nouns, as the overall experiments focus on predicting more semantically related nouns, making the results difficult to compare to Wandmacher and Antione's findings or our own findings.

While previous researchers leverage cache models to handle the differences between training and testing data, Kuhn and de Mori [47] focus on caches as a way to account for word repetition. Halliday and Hasan [35] also note word repetition as an instance of lexical cohesion. They describe cohesive relations such as lexical cohesion as the difference between a disparate collection of sentences and a cohesive text. Kuhn and de Mori [47] integrate a cache model into a trigram part-of-speech (POS) model. Their motivation for using a POS model rather than a traditional word ngram model is to allow the cache component to affect different parts of speech differently. They hypothesize that only open-class parts of speech will benefit from the cache component. In testing on the LOB corpus, they found that the trigram POS model with the cache component gave a perplexity of 107, compared to a baseline perplexity (same model without the cache) of 332.

Some researchers in word prediction have adopted more simplistic approaches to cache modeling [14, 80]. Carlberger [14] included a "recency promotion" feature for the new version of Prophet, a commercial word predictor for Swedish. They devised a recency score for each word in testing, where the value is increased by a small value for each word as it occurs (as in a traditional cache) and decreased by another small value after each sentence to decay old values. A probabilistic model was created as the product of the recency score, a trigram POS model, and a tuning parameter that was different for content and function words. This recency-adapted model was then included in the overall linear interpolation. Their large combination model is evaluated using leave-one-out evaluation on each component. They find that the recency promotion feature is responsible for 0.6% keystroke savings of the 46% keystroke savings for all components. Väyrynen [80] also implemented a simple "recency of use" component, which builds a unigram model from the most recent 400 words, excluding the most recent 2 words (in an attempt to model the repulsion factor of Beeferman et al. [4]). Unfortunately, they do not provide evaluation in terms of keystroke savings, but only hit rate, which ranges from 0% when no words are ever predicted correctly to 100% when all words are predicted at some point before the word is completed with letter-by-letter entry. The recency of mention component has similar hit rate to the best of the other models, which include word ngram and POS ngram models. However, it is difficult to draw any conclusions regarding the effect on keystroke savings.

Existing research in cache modeling indicates that cache components that account for context are more beneficial — this is apparent in Jelinek et al.'s findings (compared to their unigram cache baseline), Kuhn and de Mori's findings (compared to other researchers), and Wandmacher and Antione's findings. However, the major challenge in accounting for context in the cache model is that the data used to iteratively train the cache model is one of the most extreme cases of data sparseness, taking only the current conversation into account. Of the three implementations, Kuhn and de Mori's model requires the fewest number of parameters in the cache component — only the word and the hypothesized POS tag, compared to a word trigram in the case of Jelinek et al.'s and Wandmacher and Antione's models. Kuhn and de Mori's model provides balance between data sparseness and context, using the minimum amount of context necessary in the cache component and offloading the rest of the context onto the POS transition probabilities. Therefore we will apply Kuhn and de Mori's cache model [47] to word prediction. First we will present our preliminary findings with cache modeling in Section 4.2.1 and then outline our plans for applying Kuhn and de Mori's model to word prediction in Section 4.2.2.

4.2.1 Preliminary work

4.2.1.1 Named entity caching We implemented and evaluated the named entity cache used in Li and Hirst [57] — this method maintains a unigram model for uppercase words encountered in the current testing document and when an uppercase prefix is seen, the named entity cache populates the prediction list before the ngram model. In early tests of topic modeling, which were trained and testing on Switchboard only, we found that this approach decreased keystroke savings slightly (0.1%) The potential effect on Switchboard is relatively small as there are very few named entities in Switchboard. Additionally, since the conversations discuss a limited number of topics, the training text is adequate for predicting named entities in many cases. However, we feel that this sort of caching would be beneficial on testing text that is not similar to the training text as in our out-of-domain tests.

We originally found that Li and Hirst's named entity cache decreased keystroke savings by a small amount, however, upon re-evaluating their approach, we find that it improves performance on all corpora when using Switchboard for training. The performance improvement is roughly the same whether the trigram baseline model or the topic model is used. The results shown in Table 18 are statistically significant at $p < .0001$ for all corpora but Switchboard and significant at $p < .05$ for Switchboard. These findings are very different from Li and Hirst

	Topic		**Trigrams**	
Testing corpus	No NEC	NEC	No NEC	NEC
AAC Emails	43.527%	46.840% (+3.313%)	43.254%	46.612% (+3.358%)
Santa Barbara	43.902%	46.030% (+2.128%)	43.487%	45.638% (+2.151%)
Callhome	49.517%	52.365% (+2.848%)	49.332%	52.216% (+2.884%)
Charlotte	50.070%	52.559% (+2.489%)	49.642%	52.160% (+2.518%)
Micase	46.990%	48.998% (+2.008%)	46.524%	48.559% (+2.035%)
Switchboard*	61.478%	61.541% (+0.063%)	60.354%	60.497% (+0.143%)
Slate*	39.779%	43.029% (+3.250%)	39.174%	42.560% (+3.386%)

Table 18: Named entity caching (NEC) evaluated for the trigram baseline and the trigram topic model. All results were trained on Switchboard (for time reasons) and all corpora but Switchboard and Slate were run with cross-validation.

[57], who found a 8.5% increase in keystroke savings when trained and tested on different sections of the British National Corpus (BNC). However, Li and Hirst's evaluation was only performed on nouns, in contrast to our evaluation on fringe words (a larger set). Also, BNC is a sample of many different styles of text, so it is likely that testing on the BNC is akin to averaging keystroke savings across the different corpora we use for evaluation.

The adaptation strictly on proper nouns is similar to the component of Kuhn and de Mori's model that is used for the NNP-based POS tags, if the weight on the cache component were very high. The success of this model is additionally due to substantial variation in proper nouns between different texts. We feel that the success of this model in conjunction with the rough similarity to the proper noun component of Kuhn and de Mori's model provides additional support for an POS ngram cache.

4.2.1.2 Unigram cache backoff A simple method of testing cache-based models is to implement a unigram model of the current document, similar to the cache used to determine the topical similarity [14, 80]. When the prediction list is not filled from the ngram model (which is common when many letters have been entered or an unusual combination of letters is used), then the unigram cache is used to fill the remaining slots. In a probabilistic framework, this is comparable to holding out probability from the traditional unigram model and distributing it to the unigram cache model. This approach is similar to Carlberger's [14] and Väyrynen's [80] recency promotion.

We evaluated the baseline trigram backoff model trained on Switchboard and tested on several corpora, both with this simple cache method and without. The results are shown in Table 19. Even this simple method significantly improved keystroke savings on all corpora, both in-domain (Switchboard test) and out-of-domain (all other corpora). This simple method of cache modeling is designed so that it does not interfere with the main

Testing corpus	No unigram cache	Unigram cache	Significance
AAC Emails	43.254%	44.628% (+1.374%)	$p < 0.001\%$
Santa Barbara	43.487%	44.992% (+1.505%)	$p < 0.001\%$
Callhome	49.332%	50.734% (+1.402%)	$p < 0.001\%$
Charlotte	49.642%	50.812% (+1.170%)	$p < 0.001\%$
Micase	46.524%	49.628% (+3.104%)	$p < 0.001\%$
Switchboard*	60.354%	60.479% (+0.125%)	$p < 0.001\%$
Slate*	39.174%	40.640% (+1.466%)	$p < 0.001\%$

Table 19: Trigram baseline trained on Switchboard and tested on several corpora with and without unigram recency backoff. Cross-validation for all corpora but Switchboard and Slate.

prediction model at all, and because the predictions are always below those of the trigram model, the addition of this model is not able to decrease keystroke savings.

The success of this simple improvement demonstrates two properties of our corpora: Firstly, that there are words used in the testing corpora that are not used in the training section of Switchboard. This is true of even the testing section of Switchboard, as the model improves keystroke savings slightly on Switchboard. Secondly, this unseen vocabulary is used multiple times in the same text. Otherwise, the model would have no effect.

Even using only vocabulary adaptation, a better model would be able to integrate the vocabulary adaptations into the main predictions, so that the new words are not considered completely subordinate to the ones seen in training.

4.2.2 POS-based cache model

We plan to apply Kuhn and de Mori's cache model [47] to word prediction. The model is based on a part-of-speech (POS) ngram model , where the transition probabilities are left intact but the emission probabilities are modified to include not just the original probability from the training data, but also the dynamic cache-based probability:

$$P(w \mid w_{-1}, w_{-2}) = \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}) * ((1-\lambda) * P(w \mid tag) + \lambda * P_{cache}(w \mid tag)) \quad (6)$$

We plan to compute $P(tag \mid tag_{-1}, tag_{-2})$ and $P(w \mid tag)$ in a similar manner to our style model. We plan to compute $P_{cache}(w \mid tag)$ based on the frequency $f_{cache}(w \mid tag)$, though we may consider also including recency weighting. We will start our research into this cache model for word prediction using Kuhn and de Mori's optimal weight ($\lambda = 0.7$) and then may transition to automatic tuning afterwards. If the effect on keystroke savings is small, we will look into basing the weight on the part of speech, replacing λ with λ_{tag}, following Kuhn and de Mori's findings that a parameterized weight reduced perplexity further.

One of the problems in a POS ngram model is unknown words. Kuhn and de Mori [47] partially avoided this problem by not placing new words in any cache. However, one of the goals of cache-based language models for word prediction is to expand the vocabulary of the language model so that new words can be predicted, even if it may take several characters to make the word appear in the prediction window. There are two ways in which we can address this. Firstly, if we simply allow the probability $P(w \mid tag)$ to default to the value for unknown words, then the Viterbi algorithm should select the tag for the word that is best in the given context. This would give the tag of the new word after the sentence containing the new word is complete. We will first try this approach and investigate how well unknown words are treated. The second approach to dealing with unknown words draws on a common heuristic for POS tagging: morphology. Even a relatively basic morphological analyzer should be able to guess the POS of a word with reasonable accuracy, certainly much better than assuming that all tags are equally likely. The difference between the second approach and the first is that the first approach assumes that $P(tag \mid w)$ is a uniform distribution, whereas the second uses a distribution based purely on the word's morphology. In the first approach, the POS framework can be left intact, but for the second one, we can convert this morphology-based distribution to the emission probability using Bayes' rule:

$$P(w \mid tag) = \frac{P(w) * P(tag \mid w)}{P(tag)}$$

where $P(w)$ is a unigram model of the current document, $P(tag \mid w)$ is a morphology-based probability distribution, and $P(tag)$ is a unigram model of tags over the training data. We will perform evaluation of the first approach for tagging unknown words and if the approach is unsatisfactory, we will investigate the second approach.

A second potential problem with Kuhn and de Mori's cache model is rooted in the differences between word ngram and POS ngram models — POS ngram models do not predict collocations or semantically correct words, they simply predict grammatically correct words. Thus, a cache-based POS model is not taking full advantage of the text encountered in testing. We could address this problem by using a cache-based word ngram model following Jelinek et al. [39] in addition to the POS model. Our plan is to evaluate the cache-based POS ngram model and see if there are many words that a word ngram model would predict using fewer keystrokes. If the effect of collocations is relatively small, then we will not investigate cache-based word ngram models, but if the effect seems more substantial, we will apply cache-based methods to both a POS ngram model and a word ngram model.

We will briefly discuss one final decision in cache modeling: whether the cache model should be reset between documents or not. Our primary goals in cache modeling are vocabulary expansion and lexical cohesion. Data from other documents would hinder lexical cohesion by washing out the most recent data unless a strong decay were used. On the other hand, it would expand the vocabulary very well. We feel that the type of language model adaptation for the current (partial) document should be different than the adaptation for other documents already processed in testing. However, we feel that both provide useful information. Previous documents in testing would be useful in increasing the amount of training data and would help for out-of-domain evaluation. We feel that an actual AAC device could benefit by logging all user text and adapting to the user in general as well as the specific conversation. Although we acknowledge the potential benefit of recording all testing conversations encountered and adapting the language model to them as well as the current conversation, we will focus our current efforts on adapting to the current conversation only.

In summary, we feel that a model based on Kuhn and de Mori [47] is the best compromise between an adaptive model that takes context into account while still respecting the severe sparseness of text from the current document/conversation. Although there may potentially be some issues in using a cache-based POS ngram model, we have presented several potential solutions.

4.3 Model-specific evaluation

We have described our methodology for evaluating language models in general, but each specific language model merits specific evaluation to test whether the model is effective or not. For example, topic models are not very effective if the topic of discourse is very general, utilizing very few domain-specific terms. Similarly, when both training and testing texts focus on the same, single topic (e.g., a corpus of weather forecasts), then topic modeling will not be able to adapt much more to the topic of conversation. The most fair way to evaluate topic modeling is to ensure that the training data has multiple topics and that the testing data is such that topic adaptations have the possibility of improving keystroke savings (i.e., some overlap in topics between training and testing)[27]. Evaluation of style modeling should be similar to topic modeling — the training data should include multiple styles and ideally the testing data should include multiple styles as well. Cache-based language models have different evaluation needs than topic and style modeling. Firstly, one of the goals of cache-based modeling is to adapt the vocabulary of the language model to new words. Oftentimes, this can be evaluated by testing on a corpus not used in training. Also, the vocabulary disparity between training and testing can be validated by measuring the percentage of out of vocabulary words (OOVs) in testing. Secondly, the cohesion aspect of cache-based models can be evaluated by selecting testing data that has a very low percentage of OOVs with respect to the training data. WIth a very low OOV percentage, the evaluation will eliminate the performance contribution of vocabulary adaptation, and therefore any improvement in keystroke savings is purely due to cohesion.

We expect that performing a full domain-varied evaluation will cover all the different evaluation needs for each individual language model. The topic covered among the corpora vary widely — even within Switchboard, there are many topics, but also Micase, Slate, and AAC email are very different in topic than Switchboard, Callhome, and Charlotte. Evaluation of topic modeling can be performed by training on a topically diverse corpus (e.g., Switchboard) or training on multiple corpora. The most fair evaluation of topic modeling would be to train on the training portions of all corpora and test on the testing portions of all corpora, provided that topics are annotated in some fashion. A real-world evaluation of topic modeling could train on many different corpora and test on a corpus not used in training. Style modeling can be evaluated in the same manner, as each of our corpora is relatively homogeneous in style. The striking similarity between topic and style in evaluation is because they are both global models. Cache-based models can be evaluated using two tests. First, the model should be evaluated

[27]For Switchboard, we took measures to balance the topics across all of the cross-validation sets to ensure a fair evaluation.

on out-of-domain data, to test the OOV adaptation. Secondly, the model should be evaluated on in-domain data, to test the cohesion component. We have demonstrated that there are several complications in evaluating language models for word prediction, but we think that all significant issues have been addressed and that our evaluation techniques will provide fair tests for our language models.

5 Combining Language Models

Previous sections have described the methods we plan to apply to adapt to the current document. The development of topic, style, and cache models has presented them as independent language models. In this section, we will describe our plans for combining these three different language models to form a model that integrates all three types of adaptations. We focus on the linear combination model, which is a weighted average. We will discuss the theoretical factors in determining weights for linear combinations in Section 5.1. Then we will iteratively develop our overall combination model in Section 5.2, making note of how the theoretical factors are accounted for in practice. Finally, we will discuss specialized evaluation of the combination model in Section 5.3.

5.1 Factors in combining language models

The method of combining language models using a linear combination translates the problem of combining language models into the problem of finding a good weight for each model. This section will focus on the high-level factors in determining weights for language models.

5.1.1 Reliability

The reliability of a language model is tied to the amount of data used in training. A language model trained on more data will give better keystroke savings than one trained on less data, assuming that the type of data is the same [55]. Therefore, in combining different language models, we would like to account for the amount of training data used for each model. The combination model should focus on the most reliable models in order to produce a reliable combination model. Reliability is important when combining models trained on different amounts of text. For example, when combining a cache model and a baseline model, the cache model is trained on substantially less text, so it should be weighted less. However, the reliability of a cache model in relation to a baseline model increases over time as more words are encountered. Therefore the impact of the cache model in the overall model should increase as the cache model becomes more reliable.

Given that reliability is an important factor in weighting a language model for combination, we must find a way to programmatically determine a weight that models reliability. However, smoothing is already a way to achieve this. Smoothing methods discount a frequency distribution in proportion to its reliability. In our work, the process of using smoothing to account for reliability allows the backoff process to “focus” on the highest order ngram model that is reliable. We hope that smoothing techniques will be sufficient to work with the extreme unreliability of a cache with only a few words in it.

5.1.2 Utility

The traditional focus of weighting combination models is the utility of each language model. Our research in topic modeling applies this technique to weight the individual topic models — the utility of each topic model is estimated using similarity techniques and used as the weight in a linear combination. Similarly, Jelinek et al. [39] and Kuhn and de Mori [47] used weights in a linear combination of cache-based and static components, where the weights were used to approximate the utility of each component.

The backoff process is another example of utility being applied in existing language model combinations. Rather than using weighting, the backoff process orders language models by utility, consulting the most useful language model first and backing off to more reliable, but less useful, language models.

Utility is important when considering the difference between cache-based and static models, following previous research. We expect that with a good estimate of the reliability of each component, that the cache-based component will have a utility estimate much higher than the static component, as the cache-based component is trained using very similar data to the testing data. On the other hand, oftentimes the static training data is somewhat dissimilar to the testing data, especially in an out-of-domain test.

5.2 Combining topic, style, and cache

We think that the way in which we combine the language models will influence the success of the combination approach. Our goal in combining the language models is to arrange them so that each can contribute strong information to the overall model without washing out the information of the other models. We will present the development of the overall language model in two steps, iteratively adding language models to the overall framework. Then we will describe how other language models could be integrated with the topic, style, and cache models.

The topic model we have developed on takes the general form below (described in Section 3):

$$P(w \mid h) = \sum_{topic \in topics} P(topic \mid h) * P(w \mid h, topic) \tag{7}$$

where $P(topic \mid h)$ is estimated using cosine similarity of unigram distributions for the topic and the current document. The similarity score is improved using ITF and scaling, and stemming and smoothing when documents are used for topics. $P(w \mid h, topic)$ is computed based on the frequency of w following history h in the specified topic. However, we implemented the approach by interpolating frequencies rather than probabilities and performed smoothing and backoff afterwards, to combat data sparseness in smoothing and facilitate optimizations. The topic model takes reliability into account in three ways: Firstly, smoothing is performed on the topic model, which discounts less as the distributions become more reliable. Secondly, the model is designed so that all topics have non-zero similarity. The actual word sequences in the topic model will be identical to the word sequences in a baseline trigram model, but the probabilities will be tuned for the topic. When the topic of conversation is not identified at all, the model will degrade to the baseline trigram model. Finally, because we used frequencies in interpolation, topics are implicitly weighted for size, which helps to focus on topics that are reliable.

Utility is estimated in the topic model by making the assumption that the most similar data to the testing data is the most useful for training. The most similar topics are weighted highest and contribute the most probability to the resulting model. We feel that the assumption that similar data is the most useful is a reasonable one to make — it was one of the major factors influencing keystroke savings in our corpus study [75].

The style model takes a form similar to the topic model, but builds upon a POS ngram model (described in Section 4.1):

$$P_{style}(w \mid h) = \sum_{s \in styles} P(s \mid h) * \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}, s) * P(w \mid tag) \tag{8}$$

We envision $P(s \mid h)$ being computed in a way similar to topic, but instead of cosine similarity of unigram distributions, we will use similarity of POS unigram and bigram distributions. $P(tag \mid tag_{-1}, tag_{-2}, s)$ will be based on the frequency of the tag sequence in style s and $P(w \mid tag)$ will be computed based on the frequency of w labeled as tag. We envision the set of styles being initially determined by the corpora, because each corpus has a relatively homogeneous style. The way in which the style model accounts for reliability and relevance is similar to the topic model — smoothing and the similarity scoring implicitly model reliability and the weights model utility.

We intend to combine the topic and style model by iterating over each topic and style and determining the similarity of each to the to current conversation. Then once the similarity of each topic and style has been computed, iterating over all combinations of topic and style and using the information in a POS ngram model, illustrated below.

$$P(w \mid h) = \sum_{s \in styles} P(s \mid h) * \sum_{topic \in topics} P(topic \mid h) * \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}, s) * P(w \mid tag, topic) \tag{9}$$

The combination shows style affecting only the POS tags and topic only affecting the actual words, which we feel is the best way to model both topic and style in a POS ngram model. We feel that the Markov model approach to topic and style modeling is well-suited to this problem, allowing us to easily deal with unknown information (e.g., topic, style, POS tag) and apply the unknown information to the task.

A cache model similar to Kuhn and De Mori [47] will be added into the model (from Section 4.2).. The standard emission probability that is trained on training data is replaced by an interpolation of the standard probability with a dynamic cache-based probability:

$$\begin{aligned} P(w \mid h) = & \sum_{s \in styles} P(s \mid h) * \\ & \sum_{topic \in topics} P(topic \mid h) * \\ & \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}, s) * ((1-\lambda) P(w \mid tag, topic) + \lambda * P(w \mid tag, cache)) \end{aligned} \tag{10}$$

where $P(w \mid tag, topic)$ is the traditional component trained on the appropriate topic and $P(w \mid tag, cache)$ is the dynamic cache-based component. Initially, we will pick a generic λ value following previous researchers or using parameter tuning. If the resulting model performs poorly, we will parameterize λ on the specific POS tag, which Kuhn and de Mori found reduced perplexity somewhat. We hope to use smoothing methods to account for the relative reliability of each component. The cache-based component would be re-smoothed as it is updated, which will have the effect of increasing the reliability (and implicit weight) of the cache-based component as more of the conversation is encountered.

5.3 Evaluating the combination model

The combination of each language model can be evaluated in the same manner as the individual models. However, this evaluation is somewhat deceiving — it evaluates both the quality of the individual models as well as the quality of the combination approach. We would like to evaluate the quality of the combination approach independently of the quality of the individual models so that we can test whether the combination is good or not. We think that a gold standard for the combination model would help address this problem. The actual combination can be compared to the gold standard, showing whether the combination is falling short of our expectations or exceeding them.

The input to the combination model is a set of probability distributions, which can be combined in an arbitrary manner. Searching for an optimal function for combining the language models is impractical; the space of such functions is too large. However, we may be able to approximate an ideal combination model.

We have created an approximated ideal combination model, which we will call the *Reasonable Gold Standard (RGS)*. The Reasonable Gold Standard post-processes the results from each individual prediction method in parallel. In processing, the RGS acts like a switch, selecting the prediction method with the highest keystroke savings for each word. The RGS yields an upper bound on keystroke savings to an approximated problem — if the problem is simplified to selecting between each prediction method, the best possible combination model would select the method with the highest keystroke savings for each word (unlike the RGS, however, a real combination method would not know a priori which method will yield the highest keystroke savings).

Development of the Reasonable Gold Standard revealed another method of evaluation for combination models — an evaluation of how complementary the different components are. This evaluation is performed using two metrics:

win percentage the percentage of the time each method predicts a word using the minimum number of keystrokes of any method

exclusive win percentage the percentage of the time each method predicts a word using the minimum number of keystrokes of any method *and* is the only method to do so

These two metrics will help us to understand the degree of overlap between each prediction model. For example, we would expect the cache-based model to address some of the same words as the topic model. The exclusive win percentage of both will show whether one actually subsumes the other or whether they actually address different phenomena.[28]

6 Conclusions

We have presented our plan and existing work in building adaptive language models trained on conversational English. The adaptations focus on the topic and style of discourse, tuning the language model to better match the history of the current conversation. We will create a cache-based model which adapts directly to the document history, which both models lexical cohesion as well as expands the vocabulary of the system to be able to predict new words.

We have created a topic adapted language model that utilizes the full training data, boosting relevant portions and depressing irrelevant portions. The inclusion of all the training data as well as the usage of frequencies address the problem of sparse data while adapting to the topic of conversation. We have demonstrated that topic modeling can significantly increase keystroke savings for both traditional testing, but also when tested on text from other domains. We have also sought to address the problem of annotated topics through fine-grained modeling and found that it is also a significant improvement over a baseline ngram model. The use of topic modeling for word prediction in AAC is a novel contribution to the field. While more survey work remains to be done, the use of frequencies in topic modeling along with other approaches (e.g., stemming) to combat data sparseness for medium-grained and fine-grained topic models appears to also be a novel contribution.

We expect that this completed work will contribute a language model that is explicitly adapted to the style of discourse (both in general and for word prediction). Our application of Kuhn and de Mori's cache model [47] to word prediction will also be a new contribution. Our approach to integrating the three adaptive components will also further the field, as well as the evaluation methods for both the individual and combined models.

Acknowledgments

This work was supported by US Department of Education grant H113G040051. I would like to thank Christopher Pennington for his useful feedback and assistance with corpora and systems issues, as well as the rest of the fringe word prediction group (Amit Hetawal, John McCaw, and Debra Yarrington) for their many helpful discussions. I would also like to thank Google for subsidizing the cost of their ngram model from LDC.

[28]We hypothesize that the two will not be perfectly complementary, but that they will each contribute some unique words to the predictions.

References

[1] Gilles Adda, Michèle Jardino, and Jean-Luc Gauvain. Language modeling for broadcast news transcription. In *Eurospeech*, pages 1759–1762, Budapest, Hungary, September 1999.

[2] ANC Second Release, 2007. Accessed from http://americannationalcorpus.org/SecondRelease/ on 3/22/2007.

[3] Denis Anson, Penni Moist, Mary Przywars, Heather Wells, Heather Saylor, and Hantz Maxime. The effects of word completion and word prediction on typing rates using on-screen keyboards. *Assistive Technology*, 18, 2004.

[4] Doug Beeferman, Adam Berger, and John Lafferty. A model of lexical attraction and repulsion. In *Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain, July 1997.

[5] J. R. Bellegarda. Exploiting both local and global constraints for multi-span statistical language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 677–680, May 1998.

[6] J. R. Bellegarda. A multispan language modeling framework for large vocabulary speech recognition. *IEEE Transactions on Speech and Audio Processing*, 6(5):456–467, 1998.

[7] Jerome R. Bellegarda. Large vocabulary speech recognition with multispan language models. *IEEE Transactions on Speech and Audio Processing*, 8(1):76–84, January 2000.

[8] Adam Berger and Robert Miller. Just-in-time language modelling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 705–708, Seattle, WA, May 1998.

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

[10] Douglas Biber. *Variation Across Speech and Writing*. Cambridge University Press, 1988.

[11] Lois Boggess. Two simple prediction algorithms to facilitate text production. In *Applied Natural Language Proccessing (ANLP)*, pages 33–40, 1988.

[12] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[13] Can Cai, Ronald Rosenfeld, and Larry Wasserman. Exponential Language Models, Logistic Regression, and Semantic Coherence. In *NIST/DARPA Speech Transcription Workshop*, May 2000.

[14] J. Carlberger. Design and implementation of a probabilistic word prediction algorithm. Master's thesis, The Royal Institute of Technology (KTH), 1998.

[15] E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. Equations for part-of-speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, volume 21, pages 784–789, 1993.

[16] Stanley Chen, Kristie Seymore, and Ronald Rosenfeld. Topic adaptation for language modeling using unnormalized exponential models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 681–684, Seattle, WA, May 1998.

[17] Stanley F. Chen, Douglas Beeferman, and Ronald Rosenfeld. Evaluation metrics for language models. In *DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, Virginia, February 1998.

[18] P. R. Clarkson and A. J. Robinson. Language model adaptation using mixtures and an exponentially decaying cache. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 799–802, Munich, Germany, April 1997.

[19] Philip Clarkson and Tony Robinson. The applicability of language model adaptation for the broadcast news task. In *International Conference on Spoken Language Processing*, Sydney, December 1998.

[20] Ann Copestake. Augmented and alternative NLP techniques for augmentative and alternative communication. In *ACL-97 workshop on Natural Language Processing for Communication Aids*, pages 37–42, Madrid, July 1997.

[21] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[22] Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. The form is the substance: classification of genres in text. In *ACL 2001 Workshop on Human Language Technology and Knowledge Management*, pages 1–8, Toulouse, July 2001.

[23] C. DiMarco, G. Hirst, and M. Stede. The semantic and stylistic differentiation of synonyms and near-synonyms. In *Working notes of the AAAI Spring Symposium on Building Lexicons for Machine Translation*, March 1993.

[24] Chrysanne DiMarco and Graeme Hirst. A computational theory of goal-directed style in syntax. *Computational Linguistics*, 19(3):451–499, September 1993.

[25] S.T. Dumais, T.A. Letsche, M.L. Littman, and T.K. Landauer. Automatic cross-language retrieval using latent semantic indexing. In *AAAI Spring Symposuim on Cross-Language Text and Speech Retrieval*, pages 115–132, 1997.

[26] Afsaneh Fazly and Graeme Hirst. Testing the efficacy of part-of-speech information in word completion. In *EACL-03 Workshop on Language Modeling for Text Entry*, pages 9–16, Budapest, Hungary, April 2003.

[27] Radu Florian and David Yarowsky. Dynamic Nonlocal Language Modeling via Hierarchical Topic-Based Adaptation. In *Annual Meeting of the Association for Computational Linguistics*, pages 167–174, College Park, MD, June 1999.

[28] George Foster, Pierre Isabelle, and Pierre Plamondon. Word completion: A first step toward target-text mediated IMT. In *International Conference On Computational Linguistics (COLING)*, pages 394–399, Copenhagen, Denmark, 1996.

[29] W.A. Gale and G. Sampson. Good-Turing Frequency Estimation Without Tears. *Journal of Quantitative Linguistics*, 2(3):217–237, 1995.

[30] Nestor Garay-Vitoria and Julio Abascal. Text prediction systems: a survey. *Univ Access Inf Soc*, 4:183–203, 2006.

[31] Daniel Gildea and Thomas Hofmann. Topic-based language models using EM. In *Eurospeech*, pages 2167–2170, 1999.

[32] Jun Gong. Semantic & Syntactic Context-Aware Text Entry Methods. In *ASSETS 2007 Student Research Competition*, pages 261–262, Tempe, AZ, October 2007.

[33] IJ Good. The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, 40(3/4):237–264, 1953.

[34] Stephen J. Green. A basis for a formalization of linguistic style. In *Annual Meeting of the Association for Computational Linguistics*, pages 312–314, Newark, De, June 1992.

[35] M.A.K. Halliday and R. Hasan. *Cohesion in English*. Longman (London), 1976.

[36] Donald Hindle. Deterministic parsing of syntactic non-fluencies. In *Annual Meeting of the Association for Computational Linguistics*, pages 123–128, 1983.

[37] Eduard Hendrik Hovy. *Generating Natural Language Under Pragmatic Constraints*. PhD thesis, Yale University, March 1987.

[38] R. M. Iyer and M. Ostendorf. Modeling long distance dependence in language: topic mixtures versus dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39, January 1999.

[39] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. A dynamic language model for speech recognition. In *Proceedings of the DARPA Workshop on Speech and Natural Language*, pages 293–295, Pacific Grove, California, February 1991.

[40] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. MIT Press, first edition, 2000.

[41] Daniel Jurafsky and James Martin. *Speech and Language Processing*. Prentice Hall, first edition, 2000.

[42] Jussi Karlgren and Douglass Cutting. Recognizing text genres with simple metrics using discriminant analysis. In *International Conference On Computational Linguistics (COLING)*, pages 1071–1075, Kyoto, August 1994.

[43] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics Speech and Signal Processing*, 35(3):400–401, March 1987.

[44] Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. Automatic detection of text genre. In *Annual Meeting of the Association for Computational Linguistics*, pages 32–38, Madrid, Spain, 1997.

[45] R. Kneser and V. Steinbiss. On the dynamic adaptation of stochastic language models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 586–589, Minneapolis, MN, April 1993.

[46] H. H. Koester and S. P. Levine. Modeling the speed of text entry with a word prediction interface. *IEEE Transactions on Rehabilitation Engineering*, 2(3):177–187, 1994.

[47] R. Kuhn and R. De Mori. A cache-based natural language model for speech recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990.

[48] T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

[49] R. Lau, R. Rosenfeld, and S. Roukos. Trigger-based language models: a maximum entropy approach. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 45–48, Minneapolis, MN, April 1993.

[50] Lillian Lee. Measures of distributional similarity. In *Annual Meeting of the Association for Computational Linguistics*, pages 25–32, College Park, MD, June 1999.

[51] Yong-Bae Lee and Sung Hyon Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *Annual ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 145–150, Tampere, Finland, 2002.

[52] Gregory Lesher and Gerard Rinkus. Domain-specific word prediction for augmentative communication. In *RESNA*, pages 61–63, 2001.

[53] Gregory W. Lesher and D. Jeffery Higginbotham. Using web content to enhance augmentative communication. In *California State University, Northridge (CSUN)*, 2005.

[54] Gregory W. Lesher, Bryan J. Moulton, D Jeffery Higginbotham, and Brenna Alsofrom. Limits of human word prediction performance. In *California State University, Northridge (CSUN)*, 2002.

[55] Gregory W. Lesher, Bryan J. Moulton, and D. Jeffery Higgonbotham. Effects of ngram order and training text size on word prediction. In *RESNA*, pages 52–54, 1999.

[56] Jianhua Li. Modeling Semantic Knowledge for a Word Completion Task. Master's thesis, University of Toronto, 2006.

[57] Jianhua Li and Graeme Hirst. Semantic knowledge in word completion. In *ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, pages 121–128, Baltimore, October 2005.

[58] Milind Mahajan, Doug Beeferman, and X. D. Huang. Improved topic-dependent language modeling using information retrieval techniques. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 541–544, Phoenix, March 1999.

[59] Johannes Matiasek and Marco Baroni. Exploiting long distance collocational relations in predictive typing. In *EACL-03 Workshop on Language Modeling for Text Entry*, pages 1–8, 2003.

[60] Johannes Matiasek, Marco Baroni, and Harald Trost. FASTY - A Multi-lingual Approach to Text Prediction. In *International Conference on Computers Helping People with Special Needs (ICCHP)*, pages 243–250, London, UK, 2002. Springer-Verlag.

[61] S. E. Michos, E. Stamatatos, N. Fakotakis, and G. Kokkinakis. *Categorising Texts by Using a Three-Level Functional Style Description*, pages 191–198. IOS Press, 1996. BibTeX keeps whining about this part of a book and it won't generate the booktitle or editor.

[62] S. E. Michos, E. Stamatatos, N. Fakotakis, and G. Kokkinakis. An empirical text categorizing computational model based on stylistic aspects. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, page 71, Washington, DC, USA, 1996. IEEE Computer Society.

[63] Alan Newell, Stefan Langer, and Marianne Hickey. The rôle of natural language processing in alternative and augmentative communication. *Natural Language Engineering*, 4(1):1–16, March 1998.

[64] Fuchun Peng, Dale Schuurmans, and Shaojun Wang. Language and task independent text categorization with simple language models. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 110–117, 2003.

[65] Ronald Rosenfeld. A hybrid approach to adaptive statistical language modeling. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 76–81, Plainsboro, New Jersey, March 1994.

[66] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228, May 1996.

[67] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.

[68] Santa barbara corpus of spoken american english, 2007. Accessed from http://www.linguistics.ucsb.edu/research/sbcorpus.html on 3/22/2007.

[69] Kristie Seymore, Stan Chen, and Ronald Rosenfeld. Nonlinear Interpolation of Topic Models for Language Model Adaptation. In *International Conference on Spoken Language Processing*, Sydney, December 1998.

[70] Kristie Seymore and Ronald Rosenfeld. Using Story Topics for Language Model Adaptation. In *Eurospeech*, pages 1987–1990, 1997.

[71] Elizabeth Shriberg. Disfluencies in switchboard. In *International Conference on Spoken Language Processing*, pages 11–14 (addendum), 1996.

[72] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Text genre detection using common word frequencies. In *International Conference On Computational Linguistics (COLING)*, pages 808–814, Saarbrücken, Germany, 2000.

[73] Efstathios Stamatatos, George Kokkinakis, and Nikos Fakotakis. Automatic text categorization in terms of genre and author. *Computational Linguistics*, 26(4):471–495, 2000.

[74] *SWITCHBOARD: A User's Manual*, 2007. Accessed from http://www.ldc.upenn.edu/Catalog/docs/switchboard/ on 3/22/2007.

[75] Keith Trnka and Kathleen F. McCoy. Corpus Studies in Word Prediction. In *ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, pages 195–202, Tempe, AZ, October 2007.

[76] Keith Trnka, Debra Yarrington, John McCaw, Kathleen F. McCoy, and Christopher Pennington. The Effects of Word Prediction on Communication Rate for AAC. In *NAACL-HLT; Companion Volume: Short Papers*, pages 173–176, Rochester, NY, April 2007.

[77] Keith Trnka, Debra Yarrington, Kathleen McCoy, and Christopher Pennington. Topic Modeling in Fringe Word Prediction for AAC. In *International Conference on Intelligent User Interfaces (IUI)*, pages 276–278, Sydney, January 2006.

[78] Keith Trnka, Debra Yarrington, Kathleen McCoy, and Christopher Pennington. Topic Modeling in Fringe Word Prediction for AAC. In *International Society for Augmentative & Alternative Communication (ISAAC)*, Düsseldorf, Germany, August 2006.

[79] Harald Trost, Johannes Matiasek, and Marco Baroni. The language component of the fasty text prediction system. *Applied Artificial Intelligence*, 19(8):743–781, 2005.

[80] Pertti Väyrynen. *Perspectives on the utility of linguistic knowledge in English word prediction*. PhD thesis, University of Oulu, 2005.

[81] Horabail S. Venkatagiri. Efficiency of lexical prediction as a communication acceleration technique. *Augmentative and Alternative Communication*, 9:161–167, September 1993.

[82] T. Wandmacher and J.Y. Antoine. Methods to integrate a language model with semantic information for a word prediction component. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 506–513, Prague, Czech Republic, June 2007.

[83] Tonio Wandmacher and Jean-Yves Antoine. Training Language Models without Appropriate Language Resources: Experiments with an AAC System for Disabled People. In *European conference on Language Resources and Evaluation (LREC)*, Genova, Italy, May 2006.

[84] Tonio WANDMACHER, Jean-Yves ANTOINE, and Franck POIRIER. SIBYLLE: a system for alternative communication adapting to the context and its user. In *ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, pages 203–210, Tempe, AZ, October 2007.

[85] T.C. Witten, I.H.; Bell. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, July 1991.

[86] Maria Wolters and Mathias Kirsten. Exploring the use of linguistic features in domain and genre classification. In *EACL*, pages 142–149, Bergen, Norway, June 1999.

Appendices

A Word ngram model improvements

Word ngram models are one of the fundamental building blocks in language modeling, due to their success at modeling very short-distance linguistic phenomena. They simultaneously model syntax, semantics, and collocations within a few words. In addition to the relatively standard approach of using a trigram model with backoff as a tool for building a language model, we have experimented with a few improvements on the baseline model itself.

A.1 Dictionary backoff

When the word being typed has not been seen in training, the word prediction system will offer no advantage. In fact, the system will disadvantage the user due to the added cognitive load of scanning the word prediction list (for more details about cognitive load in word prediction, see [76]). One way to address the problem of limited training vocabulary is to augment the training with a large word list. Because the word list does not show each word in the context of a sentence, trigrams and bigrams methods are not applicable. The simplest solution is to simply populate the predictions with appropriate words from the word list if the ngram model is unable to completely fill the list of predictions. This is the approach we took, which we call *dictionary backoff*. The approach improves keystroke savings by 0.1–0.2% at window sizes 2–10 and does not change the results at window size 1. Although this method only improves the system in a small way, it cannot decrease keystroke savings and has minimal implementation costs.

A.2 Sentence-initial model

The distribution of possible words for the first word in a sentence is very different from other positions in a sentence. For example, we would expect words such as "I", "hi", and "yeah" to be more likely as the first word of a sentence than some other sentence position. A common practice in language modeling is to place a start-of-sentence pseudo-word (e.g., `<s>`) before the first word in a sentence [40, 41]. Some researchers also use an end-of-sentence symbol (e.g., `</s>`). Therefore, we created a unigram model specifically for the first word in a sentence. When this model was too sparse, the basic unigram model was used in backoff. The word is lowercased in the backoff step to reflect the typical casing of words in the middle of a sentence.

Although we have not compared the performance of using a special start token with a separate unigram model, we evaluated our baseline trigram model in early tests with and without the sentence-initial model and found that the sentence-initial model improved keystroke savings by about 5%.

A.3 Ngram model pruning

Data sparseness is a significant issue in word prediction due to the lack of substantial appropriate corpora. One method to address sparseness attempts to prune ngram models like decision trees. This method is made possible by the backoff process. The intention is for the pruning process to remove a conditional distribution if it is merely a random sample of the parent distribution. For example, if "of the X" were a very similar distribution to "the X", then recording the "of the X" distribution is wasteful. Beyond simply wasting the extra memory, however, the sample size for the more conditioned distribution ("of the X") is generally much smaller than for the less conditioned distribution ("the X"). Therefore, the more conditioned distribution is more sensitive to sparse data. However, if the more conditioned distribution is a genuine sample from the less conditioned distribution (i.e., the added conditioning word doesn't affect the distribution at all), then pruning that distribution is beneficial to combat data sparseness as well as reduce the memory requirements.

In practice, we were unable to find statistical tests to determine whether a distribution is a random sample of another distribution. Instead, we opted to use the cosine similarity between the two distributions. We can compute the similarity between the sub-distribution and the super-distribution directly, but the sub-distribution is included in the super-distribution. Because the sub-distribution can potentially contain most or all of the super-distribution, we decided to take the similarity directly between the distributions, rather than removing the sub-distribution from the super-distribution. If the cosine score exceeded a certain threshold t, the child distribution was pruned. We evaluated this approach in two settings — once with Switchboard training/testing and once with training on Switchboard and testing on Santa Barbara. For Switchboard training, we found that at $t = 0.85$, the keystroke savings was left intact, but 12.6% of the distributions of the overall backoff ngram model were removed. In our cross-domain test, we found that performance on Santa Barbara improved at most window sizes with pruning by 0.1–0.2%.

The pruning we implemented only pruned trigram distributions, but the same methodology could be applied to high-order distributions or even bigram distributions. The goal of this effort was to create a language model that could generalize and ignore conditioning information when the added conditions were irrelevant.

B Coarse-grained POS classes

Original	Reduced	Original	Reduced	Original	Reduced
CC	CONJ	NNS	N	TO	TO
CD	N	NNP	N	UH	INTER
DT	DET	NNPS	N	VB	V
EX	EX	PDT	DET	VBD	V
FW	FW	POS	POS	VBG	V
IN	P	PRP	N	VBN	V
JJ	ADJ	PSP$	ADJ	VBP	V
JJR	ADJ	RB	ADV	VBZ	V
JJS	ADJ	RBR	ADV	WDT	WDT
LS	LS	RBS	ADV	WP	WP
MD	MD	RP	PART	WP$	WP$
NN	N	SYM	SYM	WRB	WRB

Table 20: WSJ part of speech tags are shown in the left columns and the reduced POS tags are shown in the right columns.

C Runtime optimizations

C.1 Reducing adaptation overhead

C.1.1 Re-identifying the topic less often

We found that runtime performance was improved dramatically by computing the topic similarity scores less often. We decided to settle on re-scoring the topics every other sentence (and not scoring if the cache actually hadn't changed, which sometimes happens in greeting exchanges and the like). We found that keystroke savings wasn't affected much.

C.1.2 On-demand interpolation of bigrams and trigrams

Interpolation of bigrams and trigrams is performed on-demand only — as a conditioning event is encountered, the distribution associated with that event is computed (using the topic modeling linear interpolation). This saves a vast amount of computational effort, as only 1% or so of distributions occur between every similarity scoring. The results of the interpolation for each distribution are saved until similarity is re-scored to save runtime in the event that the condition is seen again. The unigram distribution, on the other hand, is always necessary, so it is interpolated right after similarity scoring.

C.2 Generic word prediction optimizations

C.2.1 Precomputed unigram lists

Unigrams are unaffected by context, so we can essentially compile down the unigram processing into mostly lookups. We presort the unigram predictions for a limited number of prefixes — the empty prefix and all prefixes of a single character. This reduces the computational overhead involved with sorting the entire vocabulary just to use unigrams in backoff. The maximum length of prefixes to precompute lists for can be tailored based on the ratio of memory to CPU limitations. While developing our baseline, we found that this significantly improved runtime performance for a small memory cost.

C.2.2 Pruning the probability computations

The probability of each word is determined by some combination of information from the probabilities in the trigram, bigram, and unigram models. In initial experiments, we computed the probability using backoff for all words in the vocabulary, then sorted the vocabulary, and selected the most likely W words, where W is the number of word spots in the prediction window. However, we quickly realized that this approach was too slow. The optimization we found was that the entire prediction list is a subset of the lists that would be generated from each method independently.[29] Therefore, the lists of possible words from trigrams, bigrams, and unigrams are sorted (in the unigram case, they are presorted) and the top W words from each of the three lists form a superset of the final prediction window. Each of these words has its backoff probability computed and then the resulting list (of maximum size $3 * W$) is sorted and the prediction list is formed from the top W words.

C.2.3 Simulating multiple window sizes at once

We simulate the prediction of words using window sizes 1–10 at the same time. A list is maintained mapping each window size to the minimum number of required keystrokes. The simulation code operates assuming $W = 10$ to start. Then after one prediction step (the list with the empty prefix), suppose the 6th word is the desired word. The number of required keystrokes for window sizes 6–10 is set to 1. Then the software reduces the desired prediction window to $W = 5$, as simulation for larger lists is no longer necessary. This process iterates until either the end of the word is reached or all size prediction windows have completed typing the word. Reducing the size of the prediction list interacts with our other optimizations, especially in Section C.2.2 where the current prediction window size is used to generate a superset of predictions to consider.

[29]This is consequence of using backoff for word prediction. Had we used some sort of interpolation of probabilities from each model, this may not be true.